

AL-TR-1991-0122

AD-A251 877



**MITT WRITER AND MITT WRITER ADVANCED  
DEVELOPMENT: DEVELOPING AUTHORING AND  
TRAINING SYSTEMS FOR COMPLEX TECHNICAL DOMAINS**

**Bradley J. Wiederholt  
Jeffrey E. Norton  
William B. Johnson  
Ellica J. Browning**

**Galaxy Scientific Corporation  
2310 Parklake Drive  
Suite 300  
Atlanta, GA 30345-2904**

**DTIC  
ELECTE  
JUN 12 1992  
S B D**

**HUMAN RESOURCES DIRECTORATE  
TECHNICAL TRAINING RESEARCH DIVISION  
Brooks Air Force Base, TX 78235-5000**

**May 1992**

**Final Technical Report for Period May 1989 - April 1991**

Approved for public release; distribution is unlimited.

**92-15246**



**92 6 10 018**

**AIR FORCE SYSTEMS COMMAND  
BROOKS AIR FORCE BASE, TEXAS 78235-5000**

**ARMSTRONG  
LABORATORY**

## NOTICES

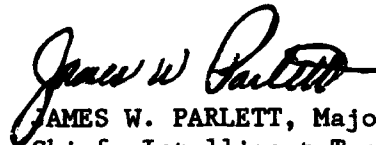
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.



KEVIN D. GLASS, Captain, USAF  
Contract Monitor



JAMES W. PARLETT, Major, USAF  
Chief, Intelligent Training Branch



RODGER D. BALLENTINE, Colonel, USAF  
Chief, Technical Training Research Division

**REPORT DOCUMENTATION PAGE**Form Approved  
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> May 1992	<b>3. REPORT TYPE AND DATES COVERED</b> Final - May 1989 - April 1991	
<b>4. TITLE AND SUBTITLE</b> MITT Writer and MITT Writer Advanced Development: Developing Authoring and Training Systems for Complex Technical Domains			<b>5. FUNDING NUMBERS</b> C - FQ7624-90-0015 PE - 63227F PR - 1121 TA - 01 WU - 01	
<b>6. AUTHOR(S)</b> Bradley J. Wiederholt Jeffrey E. Norton William B. Johnson Elica J. Browning				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Galaxy Scientific Corporation 2310 Parklake Drive Suite 300 Atlanta, GA 30345-2904			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> Armstrong Laboratory Human Resources Directorate Technical Training Research Division Brooks Air Force Base, TX 78235-5000			<b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b> AL-TR-1991-0122	
<b>11. SUPPLEMENTARY NOTES</b>  Armstrong Laboratory Technical Monitor: Captain Kevin D. Glass, (512) 536-2034				
<b>12a. DISTRIBUTION/AVAILABILITY STATEMENT</b>  Approved for public release; distribution is unlimited.			<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT (Maximum 200 words)</b>  MITT Writer is a software system for development of computer based training for complex technical domains. MITT Writer allows a student to learn and practice troubleshooting and diagnostic skills. The MITT (Microcomputer Intelligence for Technical Training) architecture is a reasonable approach to simulation-based diagnostic training. MITT delivers training on available computing equipment, delivers challenging training and simulation scenarios, and has economical development and maintenance costs.				
<b>14. SUBJECT TERMS</b> Animated flow      Graphics      Procedural advice      Time File based      Instructor feedback      Simulation Functional advice      Memory      Subsystems				<b>15. NUMBER OF PAGES</b> 50
				<b>16. PRICE CODE</b>
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UL	

## TABLE OF CONTENTS

<b>PREFACE</b> .....	ii
<b>1.0 Summary</b> .....	1
<b>2.0 Introduction</b> .....	1
2.1 Overview .....	1
2.2 Report Organization .....	3
<b>3.0 Foundations of MITT Writer</b> .....	3
3.1 Early Research .....	3
3.2 The Next Step - MITT (Microcomputer Intelligence for Technical Training) .....	5
<b>4.0 Goals for MITT and MITT Writer</b> .....	6
4.1 Candidate Domains for MITT Tutors .....	9
4.2 MITT Writer Features .....	9
4.3 System Requirements .....	11
<b>5.0 Primary MITT Writer Development</b> .....	11
5.1 MITT Writer Features .....	12
5.1.1 MITT Writer Objects .....	12
5.1.2 Translation of Objects into Tutors .....	12
5.1.3 Editors .....	12
5.1.4 User-Interface .....	13
5.1.5 Interactive Display Editor .....	13
5.1.6 MITT Writer Advisor .....	16
5.1.7 Help Library .....	17
5.1.8 Error and Consistency Checking .....	17
5.1.9 Database Support .....	18
5.2 The MITT Writer Authoring Process .....	19
5.2.1 Creating the Base Tutor .....	20
5.2.2 Demonstration and Modification .....	23
5.2.3 Delivery .....	23
5.3 MITT Writer Internal Architecture .....	24
5.4 Development of MITT Writer .....	28
<b>6.0 MITT Writer Advanced Development</b> .....	29
6.1 Minuteman Missile Message Processing Tutor ...	29
6.1.1 Fuel Cell vs. Message Processing .....	30
6.1.2 MITT Writer and MITT .....	33
6.2 MITT Writer Workshop .....	33
6.2.1 Workshop Format .....	34
6.2.2 Evaluation Results .....	34
<b>7.0 Conclusions and Areas for Future Development</b> .....	37
7.1 Conclusions .....	37
7.2 Areas for Future Development .....	37
<b>8.0 References</b> .....	38

## FIGURES AND TABLES

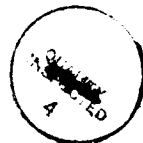
Figure 1.	MITT Writer and MITT. ....	2
Figure 2.	Sample portion of functional flow diagram for an automobile engine. ....	4
Figure 3.	Example display from the MITT Fuel Cell Tutor prototype. ....	5
Figure 4.	Creating MITT Tutors from database objects. ....	13
Figure 5.	System parts editor. ....	14
Figure 6.	Database menu. ....	15
Figure 7.	A Stack of Object Editor windows. ....	15
Figure 8.	The Interactive Display Editor. ....	16
Figure 9.	Data Field Editor. ....	16
Figure 10.	Sample Suggestion From MITT Writer Advisor. ....	17
Figure 11.	Sample display from the MITT Writer Help Library. ....	18
Figure 12.	Sample Output from Consistency Check. ...	19
Figure 13.	Overview of the MITT Writer authoring process. ....	20
Figure 14.	MITT Writer internal architecture. ....	24
Figure 15.	Minuteman Missile Tutor overview. ....	30
Figure 16.	Minuteman Missile Tutor display. ....	32
Figure 17.	Average ratings on major categories. ....	35
Figure 18.	MITT Training System scores - overall. ..	35
Figure 19.	MITT Writer Ratings - overall. ....	36
Figure 20.	Workshop Ratings. ....	36
Table 1.	Evolution of MITT Functionality. ....	7
Table 2.	MITT Functionality Terms. ....	8
Table 3.	Characteristics of Candidate Domains for MITT Tutors. ....	10
Table 4.	MITT Writer Features. ....	10

## PREFACE

The work described was supported by the Training Systems Division of Armstrong Laboratory Human Resources Directorate, under RICIS Research Activity Number E.T. 21, Subcontract Number 58 and RICIS Research Activity Number E.T. 22, Subcontract Number 59. Mr. Jim Fleming served as scientific monitor for the projects. Major Jim Parlette, Captain Kevin Glass, Captain Mike Slaven, and Dr. Wes Regian of the Human Resources Directorate monitored and supported the development of the MITT and MITT Writer systems.

Mr. Ernesto Podagrosi, MSgt Wayne Griffin, and MSgt Rick Olsen of Chanute AFB were instrumental in the development and review of the Minuteman Missile Message Processing tutor.

Our colleagues Leonard Utsman, William Pitts, and Leslie Neste assisted in the design, development, and evaluation of the MITT and MITT Writer systems.



Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## **1.0 Summary**

MITT Writer is a software system for developing computer-based training for complex technical domains. Using a training system produced by MITT Writer, a student can not only learn the system, but can also practice troubleshooting and diagnostic skills.

The Microcomputer Intelligence for Technical Training (MITT) architecture is a reasonable approach to simulation-based diagnostic training. MITT delivers training on available computing equipment, delivers challenging training and simulation scenarios, and has economical development and maintenance costs.

The MITT Writer system was developed in a 15-month effort. A workshop was conducted to train instructors to use MITT Writer. Earlier versions were used to develop an Intelligent Tutoring System for troubleshooting the Minuteman Missile Message Processing System.

## **2.0 Introduction**

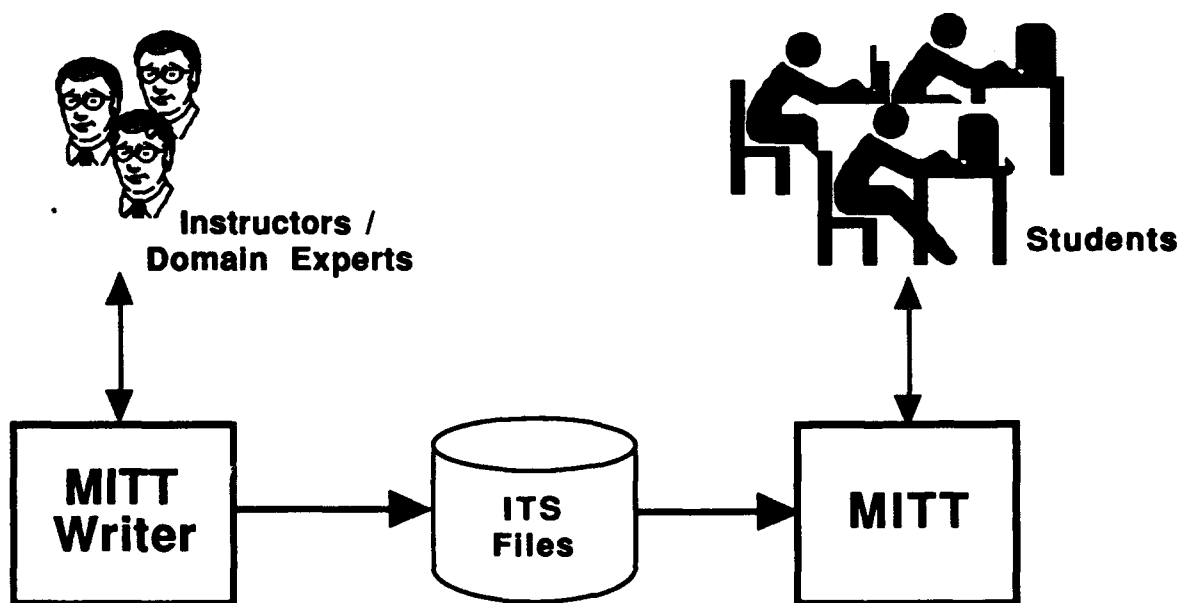
### **2.1 Overview**

MITT Writer is a software system for developing computer-based training for complex technical domains. Using a training system produced by MITT Writer, a student can not only learn about a system, but can also practice troubleshooting and diagnostic skills. In the training system, a problem is presented to a student in which a domain component has failed. The student, presented with a simulation of the system, is given initial information about the problem and then proceeds to gather additional information by viewing gauges and displays, testing components, obtaining expert procedural and functional advice and, eventually, locating the failed component.

The training system produced by MITT Writer focuses on simulation-oriented training. The training scenario relies on the simulation of gauges and instrumentation to provide the student with troubleshooting practice. During a problem session, the student receives advice that is both functional (common sense advice based on system connectivity) and procedural (based on technical orders). The student practices these troubleshooting and diagnosis techniques during several problem sessions.

Like all previous MITT projects (Johnson et al., 1985; Johnson et al., 1988; Johnson et al., 1989), MITT Writer was designed to run on conventional, readily available IBM-compatible microcomputers. Using MITT Writer, instructors and domain experts with minimal

computer experience can design an Intelligent Tutoring System (ITS)<sup>1</sup>. The ITS is made up of a system simulation, a functional knowledge base, a procedural knowledge base, a simulation interface, and instructional guidance. These components are used by MITT (a companion software program) to provide students with the new technical training (see Figure 1).



**Figure 1. MITT Writer and MITT.**

This report describes the 15-month effort in which the MITT Writer and MITT software programs were developed. During this period, a workshop was also conducted to train instructors to use MITT Writer. This report also describes earlier versions of MITT that were used to develop an ITS for troubleshooting the Minuteman Missile Message Processing System. These activities were conducted under two separate, but highly interrelated, projects: *MITT Writer* and *MITT Writer Advanced Development*.

In the first project, *MITT Writer*, the MITT Writer authoring system was developed and demonstrated. With the authoring system, a U.S. Air Force (USAF) instructor or domain expert develops Intelligent Tutoring Systems for training students to troubleshoot problems in complex technical domains. MITT Writer contains many features to support the instructor, including a graphical interface, a large help library, expert advice, and extensive error checking.

---

<sup>1</sup> General information on Intelligent Tutoring Systems can be found in (Johnson, 1988b; Massey et al., 1988; Polson & Richardson, 1988). Detailed information on the particular ITS architecture employed by MITT is given in (Johnson, et al., 1988).



In the affiliate project, *MITT Writer Advanced Development*, the evolving training tools were used to develop a turnkey ITS for the USAF. A workshop was conducted to teach prospective USAF instructors to use MITT Writer. Their initial reaction to MITT Writer was assessed (Browning, et al., 1991).

Both of these projects proceeded in parallel. Early versions of the generic MITT delivery system were used to support the development and delivery of the turnkey ITS. Likewise, the subsequent design and interface modifications to the MITT Writer authoring system were based on the reactions and opinions of the workshop participants. Internal use of MITT Writer in developing the Missile Message Processing Tutor also affected system modifications.

## **2.2 Report Organization**

The remainder of this report describes the activities and results of the *MITT Writer* and *MITT Writer Advanced Development* projects. Section 3 describes previous work conducted by the authors that contributed to the current projects. Section 4 briefly presents the technical goals for the projects, and the technical approach to meeting these goals. Sections 5 and 6 describe the key issues of the *MITT Writer* and *MITT Writer Advanced Development* projects. Section 7 summarizes the project results and offers suggestions for future research. This report does not provide detailed operation directions for either MITT Writer or MITT. A companion report, the MITT Writer User's Reference Manual (Wiederholt, 1991), provides this information.

## **3.0 Foundations of MITT Writer**

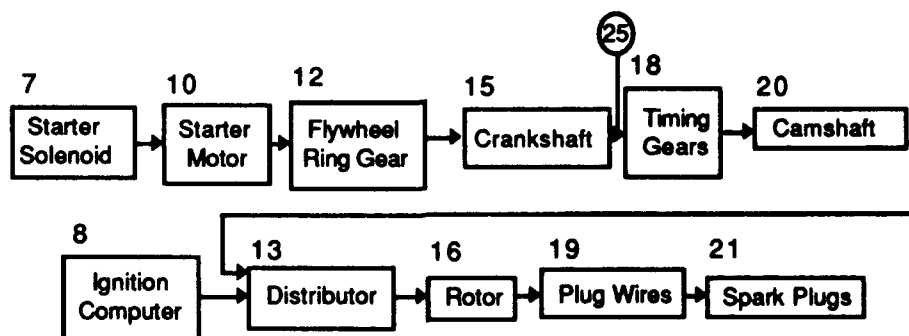
### **3.1 Early Research**

Previous efforts on part of the research team (Johnson, 1981; Johnson & Rouse, 1981; Johnson, 1987) consisted of conducting basic troubleshooting skills research and applying this new knowledge to the development of Intelligent Tutoring Systems for a wide variety of technical domains. Early research (1979-1982) focused on understanding generic troubleshooting. The first of these studies was Troubleshooting by Application of Structural Knowledge (TASK). TASK (Rouse, 1979) provided a variety of context-free simulations that enhanced the understanding of basic problem-solving. The context-free training research permitted laboratory experimentation of such variables as problem complexity, type of feedback, computer aiding, and forced pacing. The research indicated that problem-solving simulations had significant potential for "real-world" diagnostic training. This early research prompted context-specific training simulations.

Framework for Aiding and Understanding of Logical Troubleshooting (FAULT) added domain context to the problem solving sessions;

using FAULT, a student could troubleshoot a network of parts by checking instruments and making observations (Johnson 1981; Rouse & Hunt, 1984). The FAULT simulation also introduced the concept of the functional flow diagram (FFD).

The FFD taught the student to understand the concept of functional connectivity. The concept views a system as interconnected parts with "function" flowing through the system. For a part to provide proper function, all parts that connect to it and the part itself must be fully operational. This concept promotes a generic approach to all system diagnosis. A sample portion of a FFD for a car engine is given in Figure 2.



**Figure 2. Sample portion of functional flow diagram for an automobile engine.**

The basic laboratory research described above was transferred to the field with the Army SB-3614 tactical switchboard training system (Johnson and Fath, 1983 & 1984). SB-3614 examined the feasibility of using off-the-shelf microcomputers for training in the Army school environment. This effort found that low-fidelity training simulation (dictated by hardware limitations) was acceptable, but could be enhanced by practice on the real equipment (Johnson and Fath, 1984). The SB-3614 software was adopted into the Army program of instruction and was used from 1984 until 1988, when the SB-3614 equipment was retired from the Army inventory.

The concepts of functional flow and simulation-oriented training from FAULT were later applied to the Diesel Generator Simulator (DGSIM) (Johnson et al., 1985; Johnson et al., 1986). DGSIM provided improved graphics and simulation scenarios, allowing the student to 'visit' various information sources, rather than relying on the single graphical model of functional flow. DGSIM also added the ability to generate advice to the student based upon the functional flow of the system.

The DGSIM research was significant because of the extensive post-training evaluation that was conducted. The evaluation compared

the use of computer simulation to traditional training. The results showed that simulation-based training is significantly better, particularly with respect to long-term retention (Maddox et al., 1986).

### 3.2 The Next Step - MITT (Microcomputer Intelligence for Technical Training)

Work on initial versions of Microcomputer Intelligence for Technical Training (MITT) began in 1987 (Johnson et al., 1988). The first MITT prototype was a proof-of-concept software program designed to show that an effective ITS could be developed inexpensively on common and readily available IBM-compatible microcomputers. The system was based on the functionality provided by DGSIM, and featured enhancements to both the simulation and tutoring capabilities. The domain for the prototype was the Electrical Power System/Fuel Cell for the Space Shuttle. Additional features provided by this prototype included static data representation, procedural advice, Enhanced Graphic Adaptor (EGA) 640x200 pixel resolution, and unsolicited instructor feedback. Figure 3 shows an example display from the prototype.

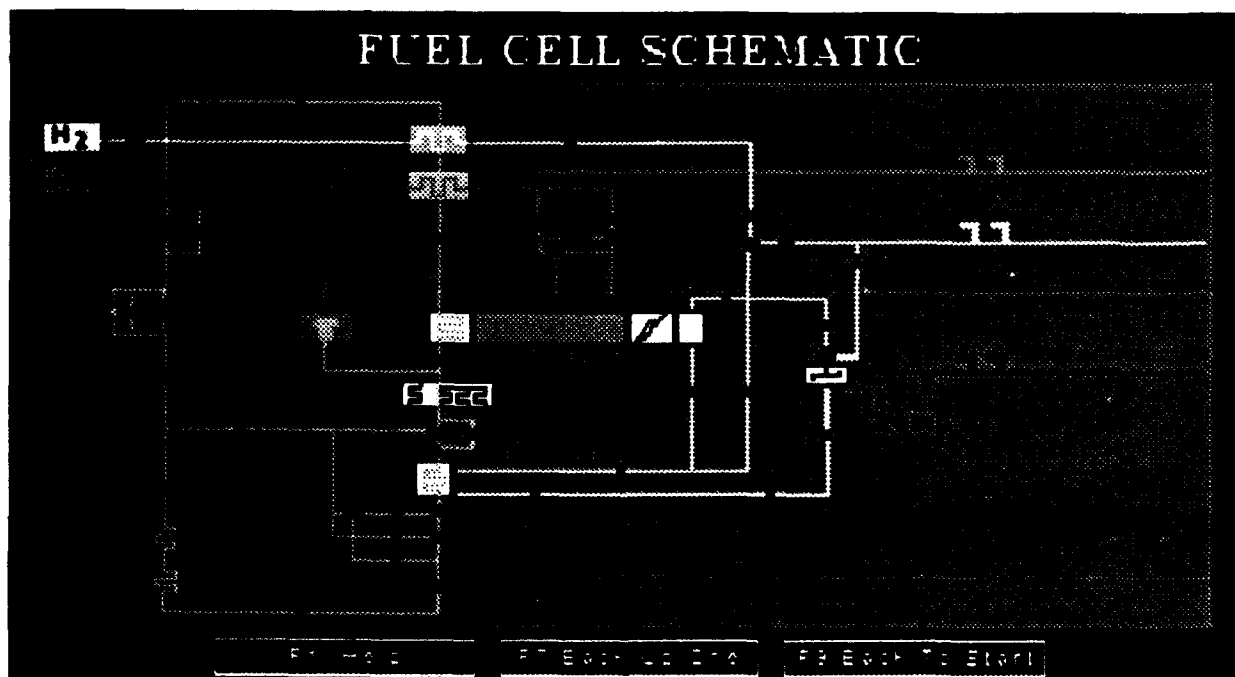


Figure 3. Example display from the MITT Fuel Cell Tutor prototype.

A subsequent MITT project enhanced the prototype to become the MITT Fuel Cell Tutor. The Fuel Cell Tutor included additional problem scenarios, dynamic data representation, an animated system schematic and overall performance enhancements (Johnson et al., 1989; Neste 1989). The Fuel Cell Tutor was evaluated at the NASA Johnson Space Center for training astronauts and flight controllers (Johnson, et al., 1988).

As an overview, Table 1 (from Norton, et al., 1991) offers a brief look at the progression of functionality provided by MITT over the past 4 years. Table 2 provides a glossary for the terminology used in Table 1.

Also during the final development cycle of the MITT Fuel Cell Tutor, a specification for an authoring system was generated (Johnson et al., 1989). With an authoring system, instructors or domain experts having minimal computer experience could develop their own ITSS. These new training systems would be based on the MITT architecture and, like MITT, would be able to run on standard USAF IBM-compatible microcomputers. Subsequent sections describe the technical hurdles that were overcome to develop such an authoring system.

#### **4.0 Goals for MITT and MITT Writer**

The initial acceptance of the Fuel Cell and Missile tutors suggested that the MITT tutor architecture is a reasonable alternative for simulation-based diagnostic training. For the MITT tutoring philosophy to be widely accepted into DOD and other government and/or industry training programs, there are three primary criteria that must be met:

- The training technology must deliver training on available computing equipment,
- The training technology must deliver challenging training and simulation scenarios,
- The training technology must have reasonable development and maintenance costs.

If MITT tutors are going to have widespread acceptance, they must operate on computer systems already installed in training installations. In the DOD environment, as well as in many other training environments, the common computer is an 80286 Intel processor-based IBM-compatible machine with, at most, 640 Kb of random access memory (RAM) and using the MS/DOS operating system. EGA color displays, combined with keyboard and mouse input, are also quite common. The MITT series of training tools was designed to operate within these constraints.

**Table 1. Evolution of MITT Functionality.**

	DGSIM	MITT Proto	Fuel Cell	Missile Tutor	MITT Core
<b>Functional Advice</b>	Includes time, most powerful, & best test	Includes only best test	Same	Same	Same
<b>Simulation</b>	System is a snapshot. Data values constant	Same	Adds simulation dynamics	Same	Same
		Adds links between mouseable regions & tests	Same	Same	Same
<b>Animated Flow</b>	None	None	Adds animation	Same	Same
<b>Procedural Advice</b>	None	Used CLIPS	Same	Expands student model	Same
<b>Graphics*</b>	CGA 4-color	EGA 640x200. Uses custom retrieval routines	Refines retrieval routines	Uses 3rd party PCX package EGA 640x350	Same
<b>Time</b>	Tracks time required to perform tests	No longer tracks time	Same	Same	Same
<b>Instructor Feedback</b>	Limited to FFD advice	Add unsolicited feedback (beyond FFD advice)	Same	Same	Same
<b>Subsystems</b>	Multiple	Same	Single	Same	Same
<b>Memory</b>	One executable	Four separate executables (Main, Info, Sim, CLIPS)	Same	Two executables Info/Sim and Clips	Overlays yield one executable
<b>File-Based</b>	No	No	No	Yes	Yes

\*CGA - Color Graphics Adaptor; EGA - Enhanced Graphics Adaptor

**Table 2. MITT Functionality Terms.**

<b>Term</b>	<b>Definition</b>
Functional Advice	The advice that is given to the student is based on the functional flow diagram (FFD). Functional advice knows only that part A is connected to part B, etc. No domain-specific advice can be given here.
Simulation	The term simulation is used here to refer to how the data values used by the system are updated. These data values can be generated in a variety of ways, ranging from the use of static data representations to the use of full-fidelity simulation.
Animated Flow	The function of the system flows from one part to the next. The flow from one part to another part can be represented with some simple animation on the screen.
Procedural Advice	Many technical training domains enlist specific troubleshooting procedures. The procedural advisor captures this procedural knowledge in the form of rules.
Graphics	The interface to MITT uses a large number of graphics. As the system has evolved, the resolution of the pictures and the manner in which these graphics are presented has changed.
Time	The amount of simulated time required to perform a test or to replace a part.
Instructor Feedback	The unsolicited feedback is similar to the feedback that would be given by a human instructor looking over the student's shoulder.
Subsystems	The size of the domain depends on two parameters: 1) the size of the executable system, and 2) the size of the domain-specific files. Sometimes the domain needs to be subdivided into several smaller systems to fit into available memory. Each of these systems is referred to as "subsystems."
Memory	The need for memory conservation is essential because of the 640 Kilobyte (Kb) memory restriction in the Microsoft Disk Operating System (MS/DOS). Several tactics were used to conserve memory during the evolution of MITT.
File-Based	In preparation for MITT Writer, all domain-specific information was removed from the code and placed into data files. In Table 1, "yes" indicates that this operation was performed, "no" indicates that the code still contained domain-specific references.

MITT tutors must be able to provide challenging, simulation-based problems to students. The architecture used for the MITT tutors is simulation-based; the student can engage in most of the diagnostic actions that are available with real equipment. This design must be available for any new MITT tutor.

The development and maintenance of a MITT tutor must be affordable. This goal can be achieved with the creation of a development environment that can be used by Government training developers and subject matter experts. Technical instructors are likely users of such development tools (Johnson, 1988a). The MITT Writer authoring environment must be relatively easy to use; it must guide technical personnel to organize their knowledge in a format that is aligned with the MITT architecture. It must provide help and advice with the development, and verify the accuracy of the newly created system.

As introduced in Section 2.0, the MITT training environment (initially proposed in Johnson, et al., 1989) consists of two separate software programs: MITT Writer and MITT. MITT Writer is a development environment used by instructors and subject matter experts to produce a description of training. In contrast, MITT delivers the instruction that was developed using MITT Writer.

The goal of MITT Writer is to support instructors and subject matter experts, rather than programmers, in developing MITT tutors. MITT Writer should provide enhanced ITS authoring capabilities. Using MITT Writer, the author should be able to develop MITT tutors and maintain them. In contrast, the goal of MITT is to deliver training to students using ITS technology. MITT should use data produced by MITT Writer to present problems, monitor student performance, compare student actions to expert actions, and furnish suitable feedback to the student. It should work with any set of description files produced by MITT Writer.

The following sections outline the technical design environment (both goals and constraints) for the implementation of the MITT Writer authoring system and the MITT training delivery software.

#### **4.1 Candidate Domains for MITT Tutors**

MITT Writer was designed to build MITT tutors for technical domains that exhibit the characteristics shown in Table 3.

#### **4.2 MITT Writer Features**

MITT Writer was designed with a variety of features to aid an instructor or domain expert in developing a new MITT Tutor. Most of these features were derived by analyzing the information needs of the MITT Writer user and the data requirements of the MITT software. Table 4 presents a list of these features. Each of these features will be discussed in more detail in Section 5.0.

**Table 3. Characteristics of Candidate Domains for MITT Tutors.**

- Complex Technical System - MITT can address a variety of interconnected components within a large system.
- System has a Variety of Instrumentation - MITT uses instruments as a basis for all testing. The more instruments that are available, the better the simulation can represent the real system.
- System Has Multiple Test Points - The MITT functional model splits the system by obtaining information at many points in the technical system. Multiple, real-world test points enhance the functional model.
- Students Must Practice to be Proficient - MITT permits continuing practice. MITT problem sessions typically tend to be short (e.g., around 15 minutes per problem), enhancing the accessibility by the students.
- Students Need Procedural Information - The MITT procedural advisor is constructed from system diagnostic procedures. If such procedures exist, and are robust, then the MITT procedural expert will be correspondingly powerful.
- Students Cannot Practice on Real Equipment - If the real equipment is simply not available for safe, effective, and efficient practice, then it is worthwhile to invest in the development of a MITT Tutor.
- Student's Job Has Diagnostic Tasks - If the job has a large troubleshooting component, the trainee must practice to learn and to sustain diagnostic knowledge and skill. If the job does not involve troubleshooting, MITT may not be the best training approach. If the job has limited troubleshooting, but such troubleshooting is critical, then MITT is also appropriate.

**Table 4. MITT Writer Features.**

- MITT Writer provides an editor for each particular training item, such as system parts, system sensors, simulation displays, diagnostic procedures, and instructor guidance.
- MITT Writer capitalizes on graphical user-interface technology using a windowed, menu-based, mouse-driven interface.
- MITT Writer provides tools for graphical screen composition and layout.
- MITT Writer contains on-line intelligent advice on how to use MITT Writer.
- MITT Writer contains an extensive help library, including context-specific help.
- MITT Writer provides on-line error checking for data correctness, completeness, and consistency.
- MITT Writer saves work-in-progress and completed work in databases, allowing authors to continue their work over several sessions.
- MITT Writer creates new MITT tutors by translating an authored database into training description files.



### **4.3 System Requirements**

One important goal of all previous MITT projects was to produce software that operates on standard, and commonly available, computing environments. MITT Writer was also designed with this goal in mind. MITT Writer also capitalizes on existing graphics tools to support its ITS authoring functions.

MITT Writer requires an IBM-compatible PC/AT MS/DOS-based platform with Intel 80286 (or higher) processor chips. Noticeable performance increases can be observed when running MITT Writer on 80386-class machines. Random Access Memory (RAM) requirements are set at 640KB or higher, although 1 Megabyte (MB) of RAM is recommended. At least 640KB of the total system memory must be conventional memory, and the remainder of the memory must conform to either the Extended Memory Manager (XMS) or Expanded Memory Manager (EMS) specifications. Noticeable improvements can be observed when the memory conforms to the EMS specifications. On AT-class machines, the addition of EMS-specific memory expansion boards may be required, although there are some software packages that can map conventional XMS memory to EMS specifications. On 80386-class machines, software (such as QEMM by Quarterdeck) is widely available to support the EMS standard.

A Microsoft (or compatible) mouse is highly recommended, but not required for using the basic MITT Writer editors. However, a mouse is required for the Interactive Display Editor. MITT Writer operates in standard 16-color EGA video mode. At least 3 MB of hard disk space should be dedicated to MITT Writer to ensure optimal performance.

MITT Writer was not designed to support bit-mapped editing of graphics. This function is left to the numerous commercial and public domain packages that are available for producing .PCX format image files. Any pictures that are produced for use in MITT Writer must be in standard EGA format (i.e., 16-color, 640x350 pixel resolution).

### **5.0 Primary MITT Writer Development**

As mentioned earlier, MITT Writer was designed with a variety of features to aid an instructor or domain expert in developing a new MITT Tutor. This section discusses these features, focuses on the MITT Writer authoring process, examines the internal architecture of MITT Writer, and discusses project development issues.

## 5.1 MITT Writer Features

### 5.1.1 MITT Writer Objects

MITT Writer allows authors to develop training systems for complex technical domains. Authors use MITT Writer to create, modify, and store objects related to the technical domain. In MITT Writer, major object classes are:

- System Components, including Parts and Sensors
- System Alarms and Faults
- Problems
- Interface Displays, including Information and Simulation Displays
- System Procedures
- Instructor Guidance

Taken as a whole, these object classes represent the Simulation, Procedural, Student Modeling, Instructional, and Interface components of a MITT tutor.

Within each of these major object types are sub-objects. For example, a Simulation Display object would have sub-objects describing controls, display elements, and active screen regions. A System Procedure would have sub-objects pertaining to the steps of the procedures and advice given during that step.

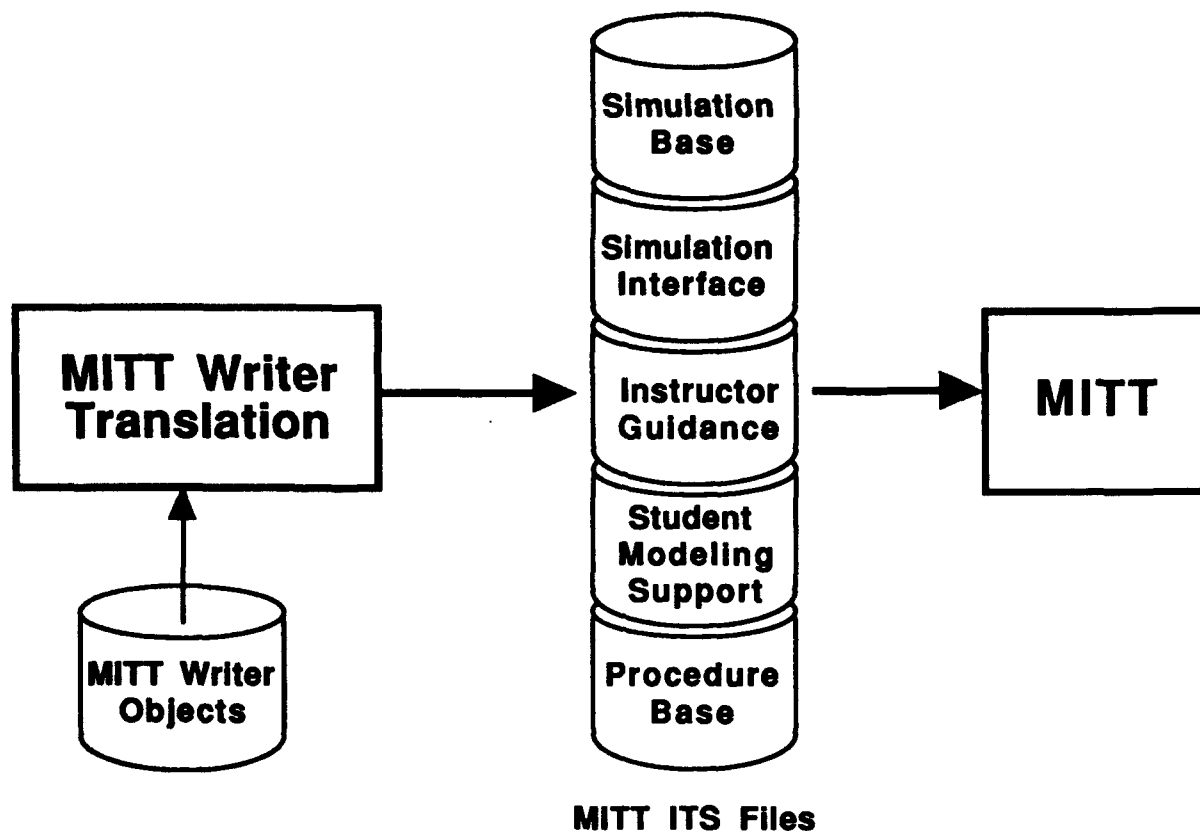
In addition to using MITT Writer to manipulate and store objects, the author can also use it to modify *relationships* between these objects. For example, system parts are related to one another based on functional flow; alarms and faults are related to particular problems.

### 5.1.2 Translation of Objects into Tutors

As shown in Figure 4, MITT Writer creates new MITT tutors by translating from MITT Writer objects into ITS (Intelligent Tutoring System) files. These ITS files are used by MITT to present the desired training to the student.

### 5.1.3 Editors

MITT Writer provides an editor for each particular training object. For example, the System Parts editor (see Figure 5) allows the author to modify the part's number, name, general description and test descriptions, functional feedback, and relationship to other parts in the system. Using an editor, the author can edit both the object and its relationships.



**Figure 4. Creating MITT Tutors from database objects.**

#### 5.1.4 User-Interface

MITT Writer capitalizes on graphical user-interface technology by using a windowed, menu-based, mouse-driven interface. This direct-manipulation style interface supports the data entry, editing, and object management tasks performed by the author. Using menus, the author can quickly select commands and tools. Figure 6 shows an example menu for all operations related to using databases.

Windows, on the other hand, logically group the author's work on the display. All work related to a particular object is located in a window for that object. Figure 7 shows a stack of windows for a variety of editors.

#### 5.1.5 Interactive Display Editor

MITT Writer provides tools for graphical screen composition and layout. For editing the Introduction, Credits, Information, and Simulation Screens, MITT Writer provides the Interactive Display Editor (see Figure 8). MITT Writer does not support bit-mapped editing of graphics. As stated earlier, this function is left to commercial packages that can produce the necessary .PCX format image files (in EGA 16-color, 640x350 pixel resolution).

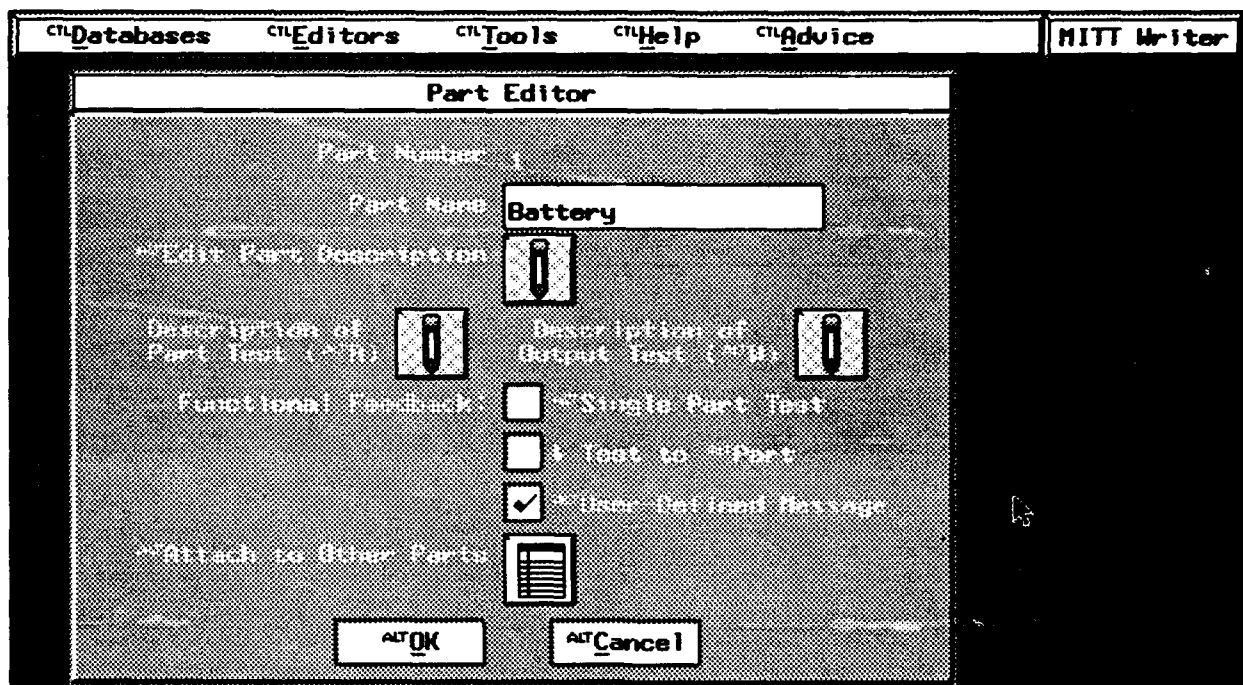


Figure 5. System parts editor.

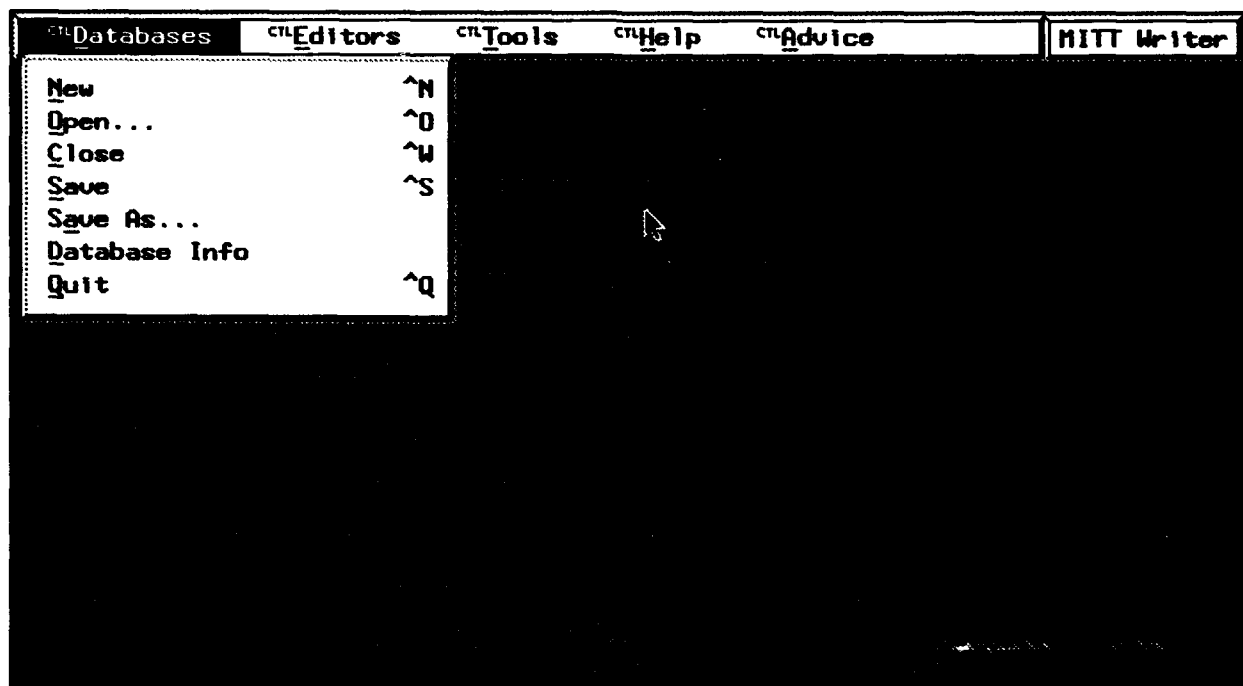


Figure 6. Database menu.

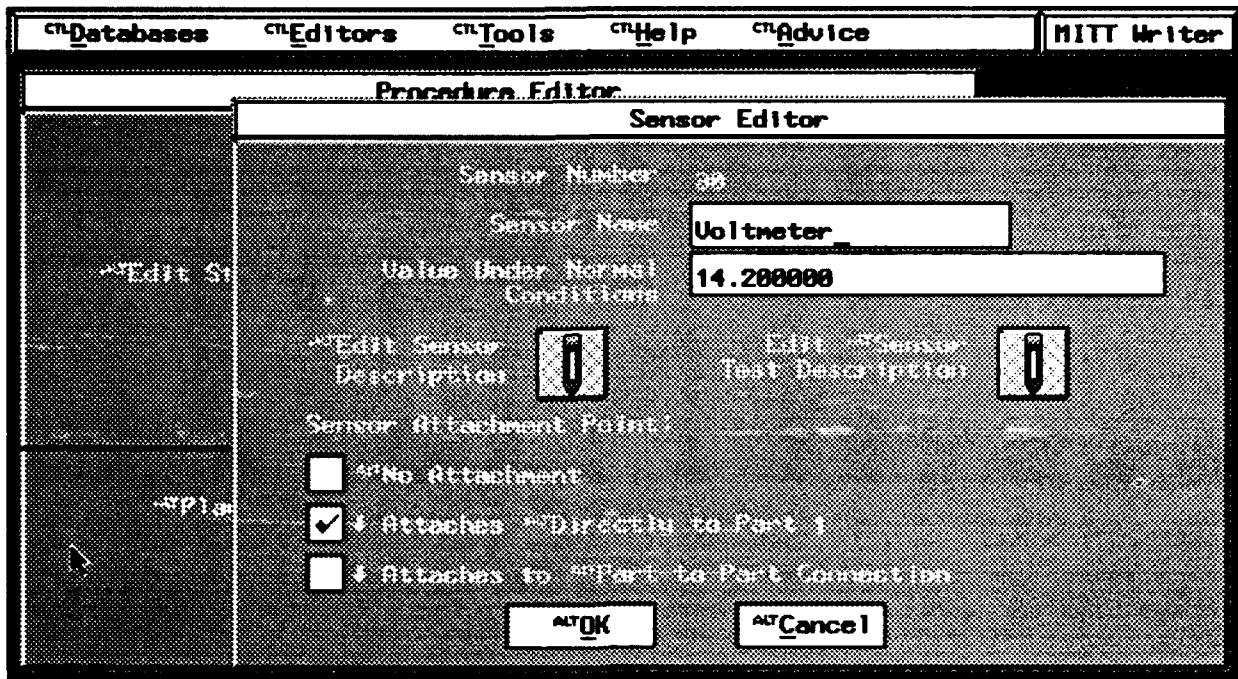


Figure 7. A Stack of Object Editor windows.

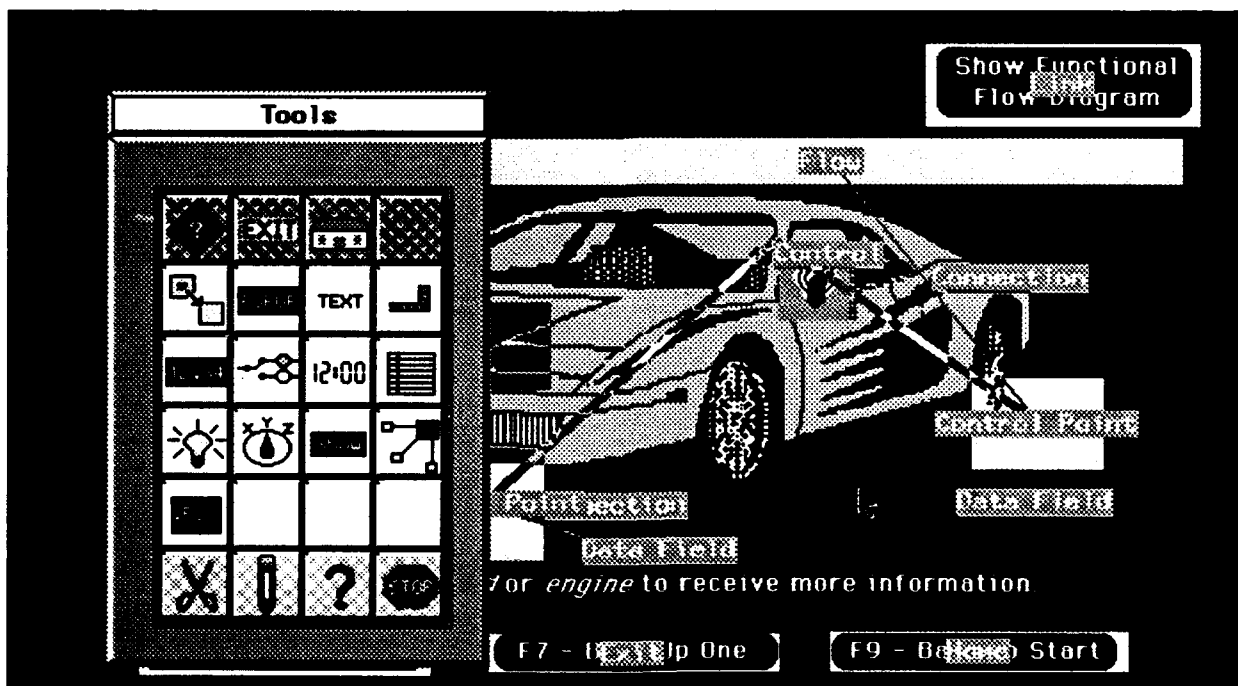
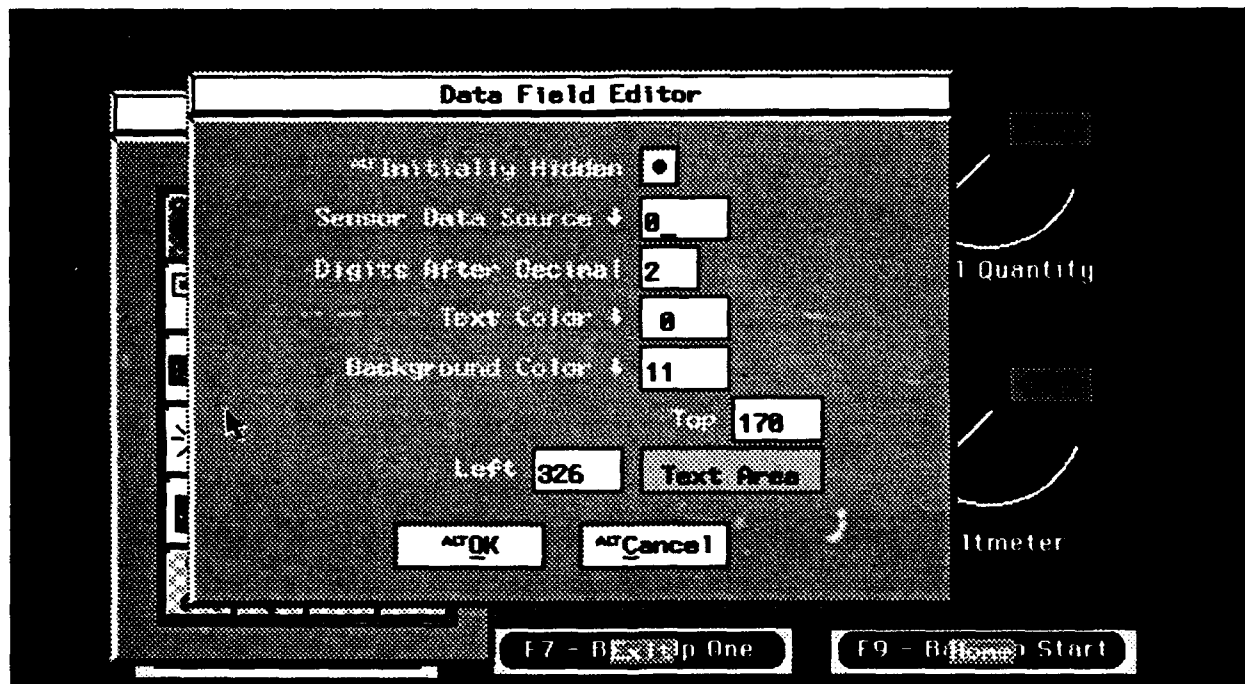


Figure 8. The Interactive Display Editor.

MITT Writer does, however, support editing and manipulating simulation- and instructional-specific display sub-objects, such as data fields, text strings, "pop-up messages," and controls. Once authors have created these display sub-objects, they can then attach them to the underlying simulation. For example, by relating a display sub-object to a system sensor object, the display sub-object inherits the value of the sensor, subsequently displaying the sensor's value to the student. (By itself, a sensor object only contains the value of the sensor. It has no ability to show this value to the student.) In the case of data fields (see Figure 9), a data field will display the value of a sensor in numeric format as the simulation runs. In the case of status indicators, the status indicator will use the sensor value to determine a status messages to be shown to the student.



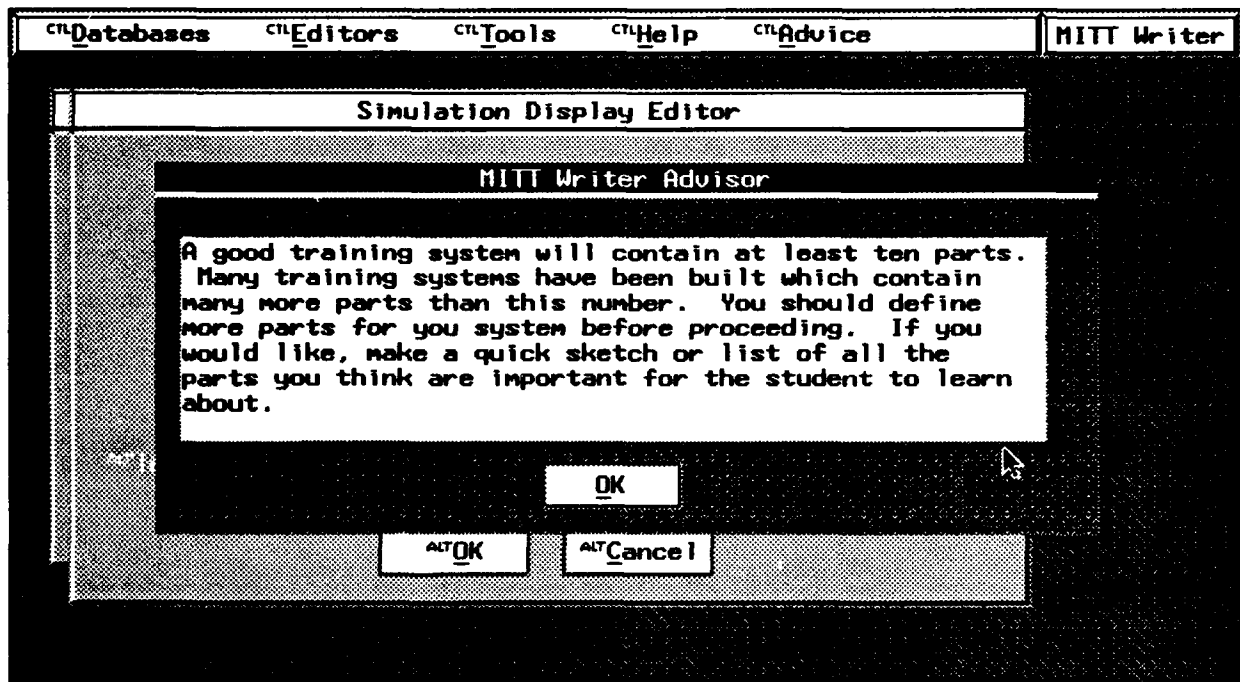
**Figure 9. Data Field Editor.**

#### 5.1.6 MITT Writer Advisor

MITT Writer contains on-line, intelligent advice to guide new authors through the MITT development methodology. This methodology has been used over the past 5 years to develop new MITT tutors. The advisor was designed to minimize the amount of off-line instruction to learn the complex authoring process.

A backward-chaining expert system was used to capture this methodology. When an author asks for advice, the Advisor will review the current database, looking for incomplete work. If necessary, the Advisor will query the author for additional

information or confirmation. After the review, the Advisor suggests the next action the author should perform, or the next object the author should create (see Figure 10).



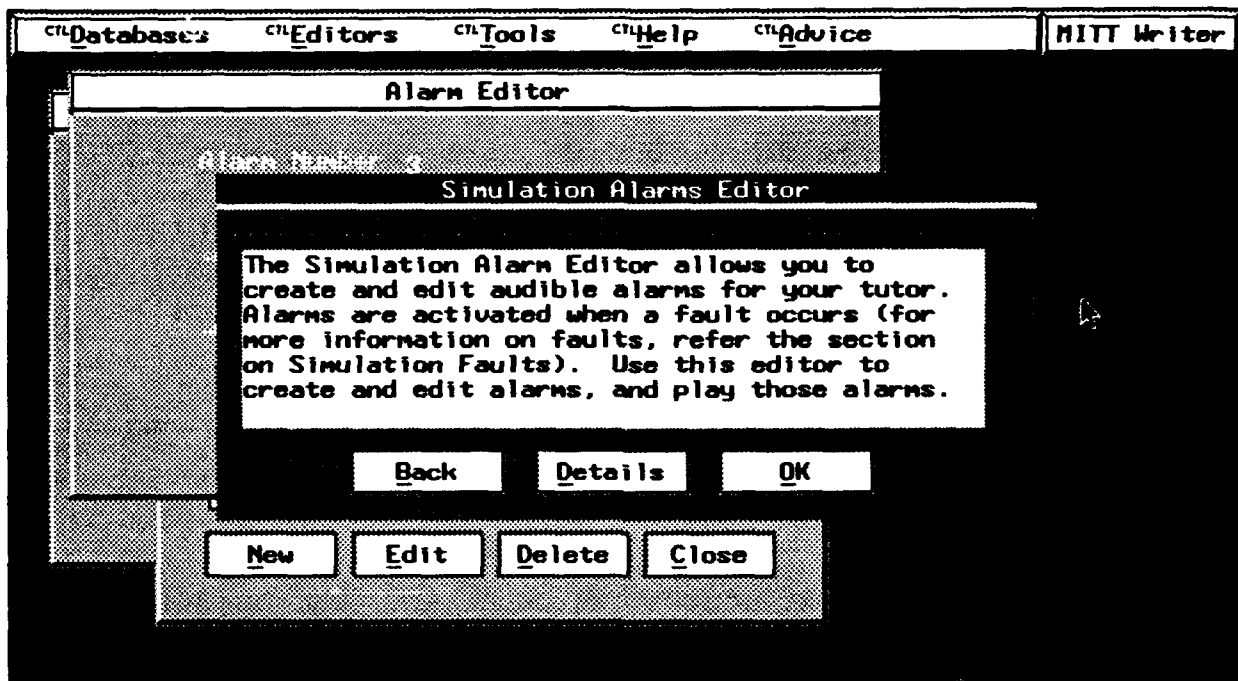
**Figure 10. Sample Suggestion From MITT Writer Advisor.**

#### 5.1.7 Help Library

MITT Writer contains an extensive help library, including context-specific help (see Figure 11). At any point during a MITT Writer session, context-specific help is available to give the author information on the object currently being edited. The help library is arranged in a hierarchical fashion to support browsing by the author. At any entry in the library, the author is able to request more detail on the particular topic, or is able to get a more general level of help. The library holds most of the material contained in the *MITT Writer User's Manual*. In fact, due to the hierarchical structure of the library, the user's manual was automatically generated from the help library.

#### 5.1.8 Error and Consistency Checking

MITT Writer provides on-line error checking for data correctness, completeness, and consistency. To ensure data correctness, validation is performed at the object level. Data that is invalid will be rejected and offered to the author for correction. To support completeness of the training system, the MITT Writer Advisor also checks the author's work.



**Figure 11. Sample display from the MITT Writer Help Library.**

To assure consistency between training system objects, MITT Writer provides a Consistency Check tool (see Figure 12). Inconsistencies occur in the database when objects reference missing objects or relationships. For example, while creating a new problem, the author might reference a procedure that has not yet been defined. The author intended to come back later and create the procedure. However, if the author forgets to create this procedure, then this 'dangling reference' will cause problems when the student attempts to run this problem. The Consistency Check tool will catch this error and present it to the author. (Note: In MITT Writer, the storage of this inconsistent reference to a procedure is not considered to be invalid. MITT Writer authors are allowed to complete their work and carry out their intentions, even though inconsistencies can occur as a result).

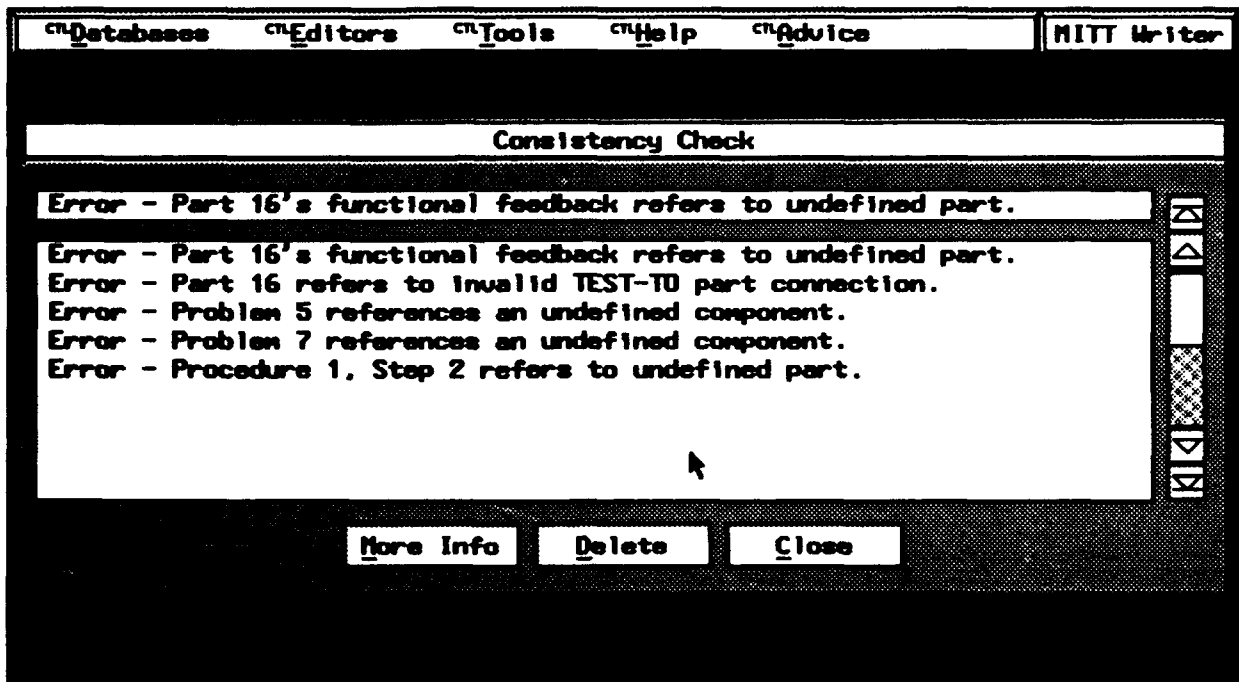
MITT Writer cannot use an inconsistent database to create a MITT tutor. Before creating a tutor, MITT Writer checks the state of the database. If inconsistencies are found, the author can correct these errors before proceeding.

#### 5.1.9 Database Support

MITT Writer authors save their work in databases, allowing them to continue development over several sessions. The development of a MITT tutor is a task requiring weeks, or even months, to complete. To support the author over these multiple work sessions, MITT Writer stores all work in databases which can be retrieved and modified at a later time. MITT Writer maintains information about



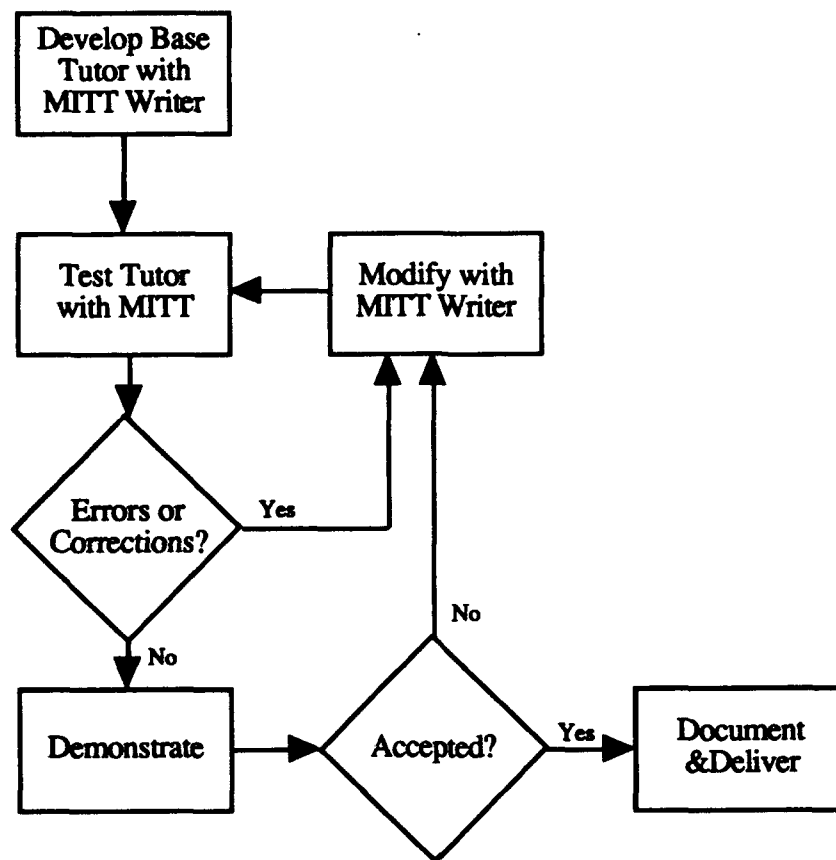
the state of the database in a special database object. Example state information includes date and time of last modification, an indication of the results of the last consistency check, and database version number. MITT Writer uses this state information to prevent the author from closing a database without first saving changes, to prevent the author from creating tutors from incomplete databases, and to maintain database compatibility with future MITT Writer releases.



**Figure 12. Sample Output from Consistency Check.**

## **5.2 The MITT Writer Authoring Process**

The MITT Writer author develops a new MITT tutor by first creating a base tutor, next testing this base tutor with MITT, and then successively refining the tutor by making additional modifications. In the past, MITT tutors were created by quickly developing the base tutor, demonstrating the tutor to potential instructors and students, and modifying the system based on their feedback. There were often multiple demonstration and feedback cycles during the development of a tutor. The development time for the base tutor is roughly equal to the time spent in modifying and retesting the tutor based on demonstration and feedback cycles. Figure 13 illustrates this development process.



**Figure 13. Overview of the MITT Writer authoring process.**

#### 5.2.1 Creating the Base Tutor

To create the base tutor, the author must pass through 5 major phases: 1) define functional characteristics, 2) create base simulation, 3) create the simulation interface, 4) develop instructional and procedural guidance, and 5) test and modify the system. Each of these phases is described below.

##### *Define Functional Characteristics*

To define the functional characteristics for the tutor, the author must first define all components (i.e., parts and sensors) for the system. This process is greatly enhanced by first sketching a rough draft of the system functional flow diagram. The functional flow is simply a graphic representation of the system's parts and sensors that shows their functional dependencies.

For each part, the author describes the part, describes how the part can be tested, and relates this part to other parts in the system. For each sensor, the author must describe the sensor, how the sensor behaves under normal conditions, and describe how the sensor can be tested; then the author must indicate the part or functional relationship that the sensor detects.

This collection of parts, sensors, relationships, and detection points indicates to the student the basic functional components and relationships of the system. The student is able to obtain information on the components, how these components are tested in the 'real-world,' and how these components are related to one another. The functional relationships also serve as a basis for the functional advisor in MITT.

#### *Create Base Simulation*

To create the base simulation, the author must define how the system acts under both normal and failure conditions. Problems are based on the failure of a component in the system. When a component fails, the system behaves differently than it would behave under normal conditions. Sensor values change, tests on components reveal abnormalities, and faults occur in the system.

MITT depends on a surface-level simulation of the system. That is, MITT simulates the observable outputs of the system. This method of simulation is adequate and appropriate for the training that MITT presents.

To create the base simulation, the author begins by defining a problem to be presented to the student. The author defines the component that will fail, the procedure that will help the student diagnose the problem, and how the values of sensors react under the failure conditions. The author defines new sensor values and the time in the simulation that sensors are set to these new values.

The author also defines faults and alarms that occur in the simulation. Faults occur when a sensor's value exceeds a predetermined range, or when a given period of time has elapsed. When a fault occurs, an associated audible and/or visual alarm is activated. These alarms are created by the author. (In MITT, faults do not cause components to fail, although this feature is a suitable enhancement for future versions of MITT and MITT Writer.)

#### *Create Simulation Interface*

In MITT, the student solves problems by reading gauges and displays, and by performing tests and procedures. The simulation interface is a series of displays that allow the student to perform these actions. The simulation interface is made up of individual displays, display sub-objects, and the relationship of these objects to the base simulation described above.

To create a simulation display, the author must first create the background graphic file for this display using an external graphics package capable of producing .PCX format graphics files. The author then uses the Interactive Display Editor to place, or overlay, MITT Writer display objects on this graphic.

MITT Writer supports several types of display objects. The author uses pop-up objects to show messages to students when they activate an area of the display; text strings to add text to the display; data fields to display the value of sensors in a numeric format; status indicators to present a text message based on the value of a sensor; flow regions to create simple animation of flow on the display; fault lists to display faults that have occurred in the system; time of day objects to display system time to the student; user defined objects to display small graphics images based on the value of the sensor; controls to allow the author to simulate toggle switches, buttons, dials, etc. (the author can hide and show display objects); show regions to hide information until the student activates an area on the display; and links to move from one screen to another.

When appropriate, the author connects display objects to sensors in the base simulation. The connected display objects use the sensor values to display a number, message, or graphic to the student.

### *Develop Instructional Guidance*

Instructional guidance falls into two categories: Procedures and Instructor Feedback. Using procedures the author guides the student through the diagnosis of a problem. The author may either generate one general procedure to guide the students through all problems, or generate a procedure specific to a given problem.

For each procedure, the author describes a series of steps that comprise the procedure. For each step, the author must describe to the student how this step should be performed, and relate this step to a student action (i.e., the author informs the student to view a sensor, test a part, or view a simulation display to perform the step). MITT uses these relationships to determine if the student has performed a step. For example, the author might determine that to perform step 3 of an engine diagnosis procedure, the student must view the value of the oil pressure sensor. The author relates this step to the oil pressure sensor. In MITT, the procedural expert uses these steps and relationships to determine when the student should perform the step, and to advise the student how to perform it.

The author can modify the Instructor Feedback component of MITT. Instructor feedback guides the student and points out features of MITT that the student might have overlooked. For example, the author might use instructor feedback to inform the student to request advice from the procedural expert (if the student has not used it during the current problem). Or, the author might also provide feedback to students if they attempt to answer the problem without gaining adequate information.

## Test and Modify System

Once authors feel that the base system is complete and ready to test, various tools are available in MITT Writer to check the database and create the ITS files. The author uses the Advisor to check for completeness of the tutor. The Advisor informs the author of missing components that were overlooked. The author also uses the Consistency Check tool to check for inconsistent database references. The Consistency Check will present errors and warnings to the author.

Once the database passes the advisor tests and consistency checks, the author creates the tutor by using the Write Training Files tool. When the author selects this tool, MITT Writer will translate the database into the necessary ITS files.

The author then uses MITT to test the training files. In this mode, the author plays the part of the student: running problems, getting help and advice, viewing gauges and displays, and testing parts. In many cases, the author will notice errors and will note possible modifications to the tutor. In these situations, the author returns to MITT Writer, and makes the desired corrections.

### 5.2.2 Demonstration and Modification

After the author has created the base tutor, demonstration to outside parties (both instructors and students) is necessary to obtain additional perspectives on the design, useability, and acceptability of the tutor. In most cases, the initial demonstration of the system yields feedback that often points to critical modifications that need to be made to the tutor. These modifications often center upon the addition of functionality (e.g., more simulation screens, more parts and sensor points). When developing a MITT tutor, the author should plan on conducting multiple demonstration, evaluation, and modification cycles. As mentioned earlier, the time spent during these cycles is often equal to the time spent developing the base tutor.

### 5.2.3 Delivery

After developing the tutor, and before delivering and distributing the new MITT training system, the author must create supporting documentation for the system. At a minimum, the author should create a student manual for the tutor. Template student manuals are provided to the author in the *MITT Writer User's Manual*, and on the MITT Writer distribution diskettes. The author modifies these templates by adding domain-specific information (e.g., functional flow diagrams).

MITT Writer supplies utility programs for 'bundling' the new MITT tutor. Using these tools, the author can copy all of the necessary ITS and graphics files to distribution diskettes. The author delivers these diskettes, the MITT delivery diskette, and the newly created manual to prospective instructors and students,

but may choose to retain or distribute the authoring database. There are utility programs available to aid the author in this distribution process.

### 5.3 MITT Writer Internal Architecture

MITT Writer's internal architecture is depicted in Figure 14. There are 7 major components of the MITT Writer system: 1) the Display Manager, 2) the Database Manager, 3) Object Editors, 4) the Consistency Checker, 5) the Help System, 6) the Advice System, and 7) MITT File Support. Technical and implementation details for each of these components are discussed in this section.

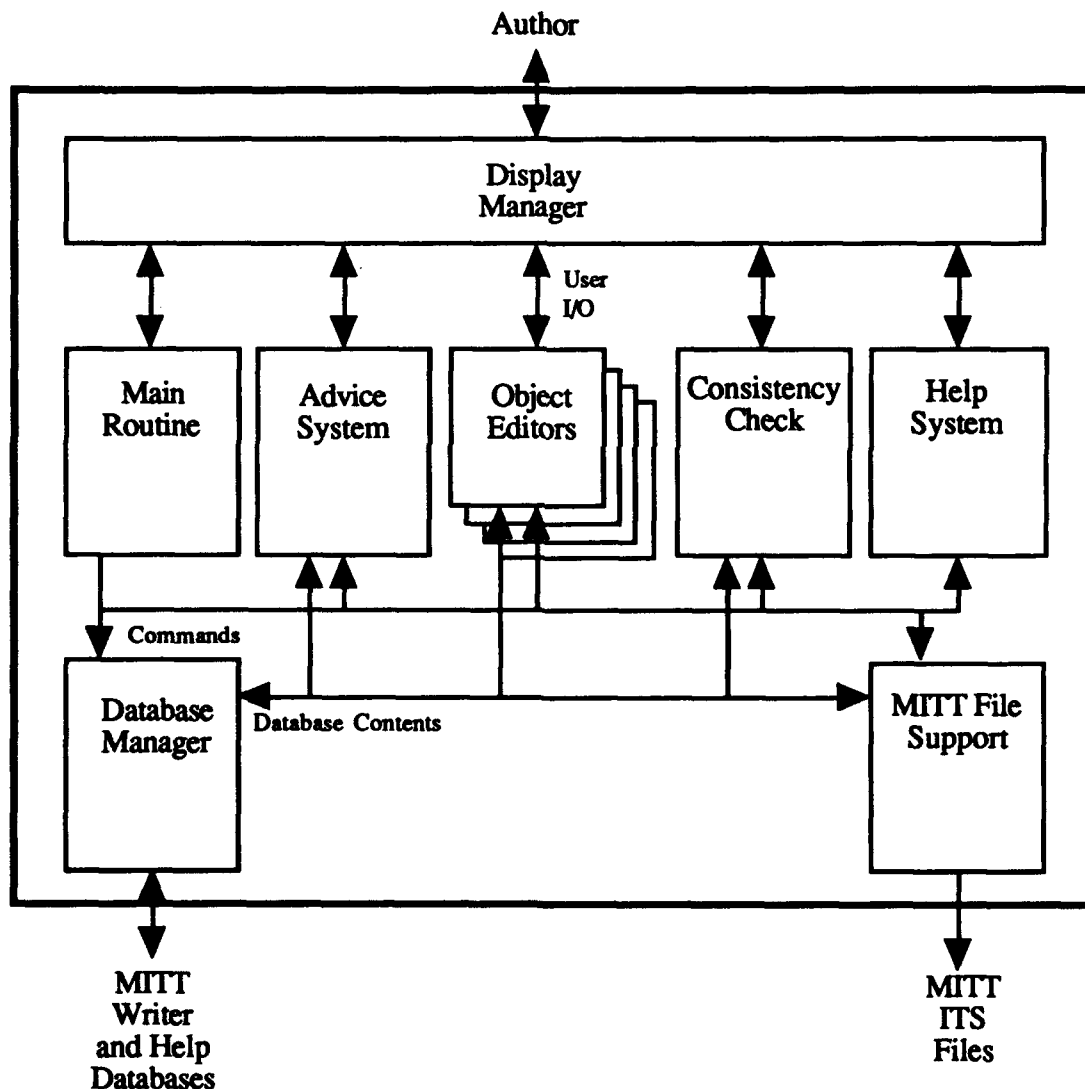


Figure 14. MITT Writer internal architecture.

## *Display Manager*

The Display Manager centralizes all interface management routines and serves as an intermediary between the author and the MITT Writer system. It maintains information about the display state of MITT Writer, opens and closes windows, and processes all author actions.

The Display Manager was implemented using a *message-passing* design. When the author makes a gesture (i.e., moving the mouse, selecting a display item, pressing a key), the Display Manager first determines if the gesture should be handled by the Display Manager or by external routines. In the case of low-level gestures (e.g., moving the mouse), it is capable of processing this gesture alone. However, in the case of high-level gestures (e.g., the user closing a window), the Display Manager must work with external routines to close the window, refresh the screen, and activate the next window. For these high-level gestures, the Display Manager passes an instruction message to the appropriate window(s), giving them directions.

For example, in the case of closing an editor, the Display Manager activates the editor first to save the object being edited, and then to close the editing form. The Display Manager then closes the window. The Display Manager next sends messages to background windows, directing them to refresh their respective portion of the screen. Finally, the Display Manager sends an activate message to the new top window.

To support the graphical interface, the Display Manager depends on two third-party graphical libraries. Low-level graphics are supported by Metagraphics' *Metawindows*. High-level window, frame, and menu processing is supported by Ithaca Street Software's *Menuet*.

Because of the advanced functionality provided by the Display Manager, this component of MITT Writer consumes the most computing resources, both in storage (on disk and in memory) and in CPU cycles (for screen refreshes, message handling, etc.). The memory requirements for the Display Manager, along with those for the object editors, strained the MS/DOS 640KB memory limit. As a result, the Display Manager was heavily optimized during the later stages of MITT Writer development.

## *Object Editors*

From the MITT Writer author's point of view, the object editors provide the majority of the functionality provided by MITT Writer. These editors are the most heavily used components of the system. The basic design for each editor is that of a frame. The author is presented with the frame for an object, and enters information via the frame into the object structure. In the case of the Interactive Display Editor, this frame concept was augmented with direct-manipulation to relieve the author from having to

understand the concepts of screen coordinates, and subsequently entering them into the frame for a display object.

The object editors interact heavily with both the Display Manager and with the Database Manager. The object editor is responsible for using the Display Manager to display object information on the screen, and for validating data before using the Database Manager to store the object in the database. As a design consideration, the object editors also use two graphics libraries to manage frames and manipulate graphic images.

### *Database Manager*

The Database Manager was designed to support the permanent storage of objects and their relationships, and to maintain information about the state of the database in use. Components in MITT Writer communicate with the Database Manager via objects and functions. Objects are passed to and received from the Database Manager. Database Manager functions are available to control and manipulate these objects in the database.

In addition to the database used by the author, the Database Manager also controls the System Message database. The System Message database is used to hold all text necessary to support MITT Writer. Example text includes error messages, help text, and advice presented by the Advisor. These items are stored in an external database to conserve space in system memory.

MITT Writer components require both sequential and random access to the objects stored in the database. For example, the consistency check tool performs a sequential search on the database, scanning each object. Object editors, on the other hand, require random access based on an identification code for the object (e.g., part number). The Database Manager supports both access methods.

The Database Manager was implemented using a B+tree as the file storage and indexing method. A commercial library, *CBTREE* by Peacock Systems, was used to support the low-level B+tree format. A network database structure was implemented on top of this format. This approach required minimal memory overhead, though the required disk space is comparable to that of a commercial database management system (DBMS). Commercial DBMSs that support either network or relational models were considered, but were rejected because of high RAM requirements.

### *Consistency Checking*

The consistency checking tool scans the database and searches for references to missing objects. A list of inconsistencies is produced at the end of this scan. Two levels of inconsistencies can occur. Warnings are inconsistencies that degrade the operation of the tutor, but do not prove fatal. However, they should be corrected. For example, a simulation display may



contain a link object that references itself (i.e., when the student activates this link, the same display is shown). MITT is able to handle these warning-level inconsistencies. Errors, on the other hand, are inconsistencies that prohibit the MITT tutor from operating properly. For example, if a functional connection leads to a missing part, the MITT functional advisor will give improper advice. MITT Writer tags these inconsistencies as errors.

In addition to maintaining a list of inconsistencies in the current database, the consistency checking tool also provides additional details for each inconsistency. If authors wish to see additional information regarding an inconsistency, they select that inconsistency and ask for more information. The consistency checking tool then provides additional details and offers advice on how to correct the inconsistency.

### *Help System*

As mentioned earlier, help text is stored in the System Message database. Besides storing individual help entries, the database also stores the relationships between these help entries. Using these relationships, the author can browse through the help database to obtain either a more detailed or a more general level of help for a given entry.

The help system contains both orientation help (e.g., how to use MITT Writer) and context-specific help (e.g., how to enter data into the current object editor). Context-specific help is supported by tagging each object editor with the identification code of the corresponding help entry. When the author asks for help, the help system presents this entry. This same concept could be extended to individual object components, but was not applied in MITT Writer.

To generate the *MITT Writer User's Manual*, a program was written to print the contents of the help database. Given a starting entry, the program traversed the database structure, formatted, and printed the manual to disk. Figures were then added to this file. Using this technique, future changes to the help database can be easily reflected in the user's manual.

### *Advice System*

MITT Writer contains an embedded expert system that guides the author through the MITT Writer authoring process. To develop this advisor, a prototype version was implemented using the 'C' Language Integrated Production System (CLIPS) expert system shell. The expert system is implemented as a backward-chaining system, with the advisor attempting to assert that the author's database is complete. The MITT authoring methodology and previous experiences in developing tutors supported the creation of the initial rulebase. After review and modification of the CLIPS version, the system was converted to a 'C' language based system. Again, memory constraints dictated this approach.

As the MITT Writer Advisor runs, it scans the appropriate sections of the database (as dictated by the rules). The advisor also queries the author to gather information that cannot be inferred from the database. When the advisor reaches a point at which it cannot continue, it offers the advice associated with the expert system's current sub-goal. For example, if the advisor is trying to reach the sub-goal of "All parts defined," then the fact that there are no parts in the system will generate an advisor message. Similar to the help system, the advisor retrieves necessary messages from the System Message database.

#### *MITT File Support*

The MITT File Support component of MITT Writer translates an author's database into a format suitable for MITT. Translation is necessary because of the different file formats required by the MITT tutor versus those required by MITT Writer. A database that has passed the consistency check with no errors is eligible for translation.

To translate the database, the MITT File Support component creates each file needed by MITT individually. The database is both randomly and sequentially read during translation. Disk I/O requirements for the translator are extensive, contributing to excessive execution time for this tool. Modifications were made to reduce the number of disk reads. In addition, to appease the waiting author, an estimation of "percent complete" is displayed during translation.

#### **5.4 Development of MITT Writer**

The MITT Writer Development project spanned 15 months, and focused on two major tasks. In one task, the existing MITT Fuel Cell Tutor had to be modified to create a generic training system shell that could be used with any domain. This task involved examining the code for items that, if changed, could be reused to represent a new domain. Data files were defined for these items. The MITT software was then modified to read these files and present the prescribed training.

The other major task involved designing and implementing the MITT Writer system. From the start, previous experiences in developing other MS/DOS-based systems led us to believe that MITT Writer would challenge the capabilities and resources of the target computing platform. Subsequently, the technical approach to the design of MITT Writer focused on maintaining desired functionality while conserving computing resources, which became a critical issue when choosing a development language, surveying third-party software, and developing code.

During development, the early versions of MITT Writer and MITT were designed to support the development of the Minuteman Missile

Message Processing Tutor (described in Section 6.1). As such, early emphasis in the project was directed at making MITT as generic and file-driven as possible to support this new domain. Early versions of MITT Writer's file generation capabilities were used to create these new files.

Early MITT Writer development focused on designing and implementing the Display Manager and Database Manager. The majority of the Object Editors were added next. To complete the base MITT Writer system, the Consistency Check tool and complete file translation capabilities were added. Also during this period, enhancements were being incorporated into MITT, such as optimizing the procedural expert, embellishing portions of the student interface, and implementing support for additional display objects.

The base MITT Writer system was demonstrated in October, 1990. During this demonstration, a sample database was modified, checked for consistency, and translated into MITT ITS files. The MITT software then used these files to present the modified training.

Between October 1990 and February 1991, the consistency check system was enhanced, and the MITT Writer Advisor and Help System were incorporated. The MITT code continued to improve as support for user-defined objects and controls was implemented, and as enhancements were made in the student interface and simulation control functions.

The MITT Writer and MITT systems were first used by outside parties during the MITT Writer Workshop (see Section 6.2), held in early February 1991. In this two-day workshop, the participants used MITT Writer to modify an existing tutor and to test their modifications using MITT. Suggested interface enhancements and programming changes were collected during the workshop. These suggestions were prioritized and used to guide the final development.

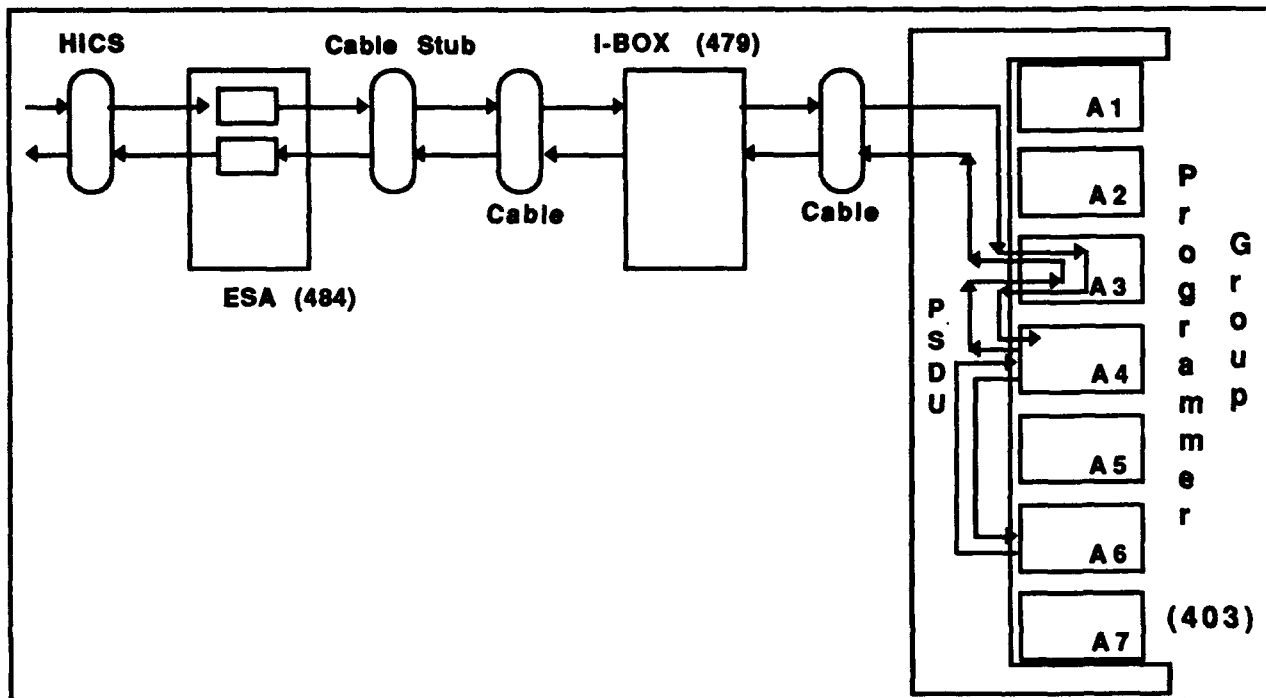
During late February and March 1991, the MITT and MITT Writer systems were completed and delivered in April 1991 along with updated versions of the two user's manuals.

## **6.0 MITT Writer Advanced Development**

### **6.1 Minuteman Missile Message Processing Tutor**

The first part of the Advanced Development project used the MITT paradigm to develop an Intelligent Tutoring System for an operational training unit of the USAF Air Training Command (ATC). The domain was identified as the Message Processing System for the Minuteman missile. Galaxy Scientific personnel worked closely with personnel from the Chanute AFB Technical Training Center to develop the system.

The Message Processing System for the Minuteman Missile involves data communications within a network of launch facilities and launch control facilities. A message enters the site through a hardened cable, is processed through various communications drawers, and is passed on to other sites, as shown in Figure 15. The Minuteman Missile student must ensure that this signal is valid upon entering the site, is processed properly, and is routed to the next site.



**Figure 15. Minuteman Missile Tutor overview.**

#### 6.1.1 Fuel Cell vs. Message Processing

The original design of MITT (developed in cooperation with NASA) was shaped by the Electrical Power System (Fuel Cell) on the Space Shuttle. The Fuel Cell had many properties that heavily influenced MITT's capability. This section will compare these properties with the properties exhibited by the Message Processing System and explain how the differences were accommodated.

##### Gauges

The first difference involves how data is presented to the student in each system. The Fuel Cell presents many video displays and gauges to the astronauts and flight controllers. To receive information about the system, the astronaut has only to look at a display or gauge. In contrast, the Message Processing System has very few instruments, but continuously presents data to the student. The Minuteman Missile student is forced to decide not

only what equipment to look at, but also how to look at the equipment.

For example, if astronauts suspect trouble with the fuel pump, they simply look at one of the displays or gauges to check the value. In contrast, if a Minuteman Missile student suspects trouble with a drawer, a connector must be removed and attached to a piece of monitoring equipment (a headphone or volt meter). This tasking demands a much more active role on the part of the Minuteman Missile student, which in turn creates greater software requirements.

#### *Data Entry Support*

This more active role requires the Minuteman Missile student to not only choose the component to inspect, but also to choose the monitoring equipment and how and where to attach it. The simple Fuel Cell gauge interface had to evolve to support this more dynamic data entry procedure.

An example scenario will help to explain the data entry procedure. The Minuteman Missile student monitors equipment several different ways. Assume that the system contains only four pieces of equipment: Drawer A, Drawer B, Drawer C, Drawer D. Also assume that only four tests are available to monitor the equipment: Test 1, Test 2, Test 3, and Test 4. In a realistic scenario, the student would perform each test on each drawer which requires support of 16 different tests for this small example. If each test required that the student identify two different pins on each drawer for each test, the number of actions that must be supported would grow exponentially.

If a programmer were modifying the system to support this test cycle, the programmer could use very specific statements, similar to the following:

"If Test=1, Drawer=C and Pins=2,5  
then display a data value of +12 Volts"

However, to support this kind of logic in an authoring system requires the development of a scripting language. The authoring system for MITT does not support scripting, so another solution had to be used.

#### *State Machine over Scripting Language*

The solution treats the system as a state machine. Each screen that the student sees represents a certain state. For example, the fact that the student sees Screen 2.3, implies that the student has already seen Screens 2.1 and 2.2. This state machine eliminates the need to keep track of global variables such as Test, Drawer and Pins.

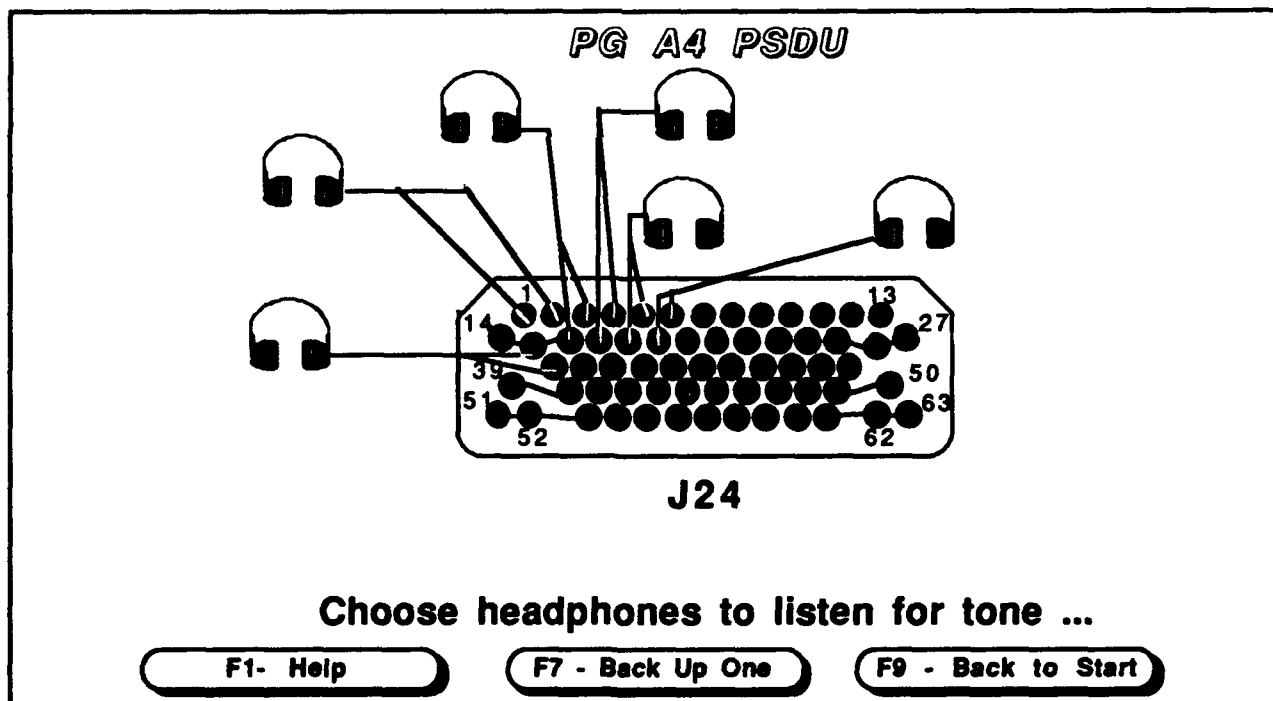
The basic MITT "Gauges" interface could support this more dynamic data entry procedure, but at a cost in memory and data size. As mentioned before, each state of the system must be represented by

the system. Since the combinations of equipment, tests and pins is rather large, the amount of data required to represent these combinations is also large. However, this approach agreed with the MITT Writer Authoring System design.

### *Procedures*

The level of detail required to represent procedures (not procedural advice) differs between the Fuel Cell and the Missile Tutors. In the Fuel Cell, the astronaut only needed to know which procedure to perform. The details of how to perform the given procedure were not important. The astronaut merely chose an item from a list of procedures.

In contrast, the Minuteman Missile student was concerned with identification of the correct procedure, as well as exactly how to perform the procedure. Accordingly, the Minuteman Missile student was required to select the procedure to perform, the piece of equipment on which to perform the procedure, and the pins to which to be connected to the test device. The last of these steps is shown in Figure 16. Therefore, as previously stated, a more active role was required of the Minuteman Missile student than of the astronaut.



**Figure 16. Minuteman Missile Tutor display.**

### *Interface Suggestions*

As suggested by the intensive nature of data entry, the data entry portion of the Minuteman Missile system (Procedures/Indications

selection from the Options Menu) becomes the primary focus of the system. This focus emphasizes the procedures and de-emphasizes the use of the functional flow of the system. ATC personnel suggested that the primary screen be the Procedures/Indications screen and that the other functions listed on the current options menu be included in a "help" key. An answer button could also be added on the main screen. The "help" function would then list Functional advice, Procedural advice, remaining parts, and information and description functions.

While this addition seems to make sense for this domain, it is questionable whether it would make sense for all domains. Also the de-emphasis on the functional flow is not desirable. However, the MITT tutor was enhanced to appear less text-based and more graphical.

#### 6.1.2 MITT Writer and MITT

As mentioned earlier, the Minuteman Missile Tutor was designed with MITT Writer in mind. Every effort was made to guide the tutor design to be compatible with MITT Writer. Occasionally, the tutor development uncovered some MITT Writer design deficiencies.

The Minuteman Missile system was developed by programs to generate the files that MITT reads. Even though MITT Writer was not mature enough to support the full development of the system, it required only the following three custom software changes of MITT:

- removed the sensors from the feasible set because sensors are only used as data sources, not components for the Minuteman Missile System
- suppressed some facts from being asserted in CLIPS to conserve memory
- changed the options menu from "Gauges G" to "Procedures/Indications N"

The Minuteman Missile domain was a very challenging one. It required extensive analysis and planning for incorporation into the existing design. ATC personnel also pointed out several areas where the design of the MITT interface could be changed to make more sense in the new Procedures/Indications context. While certain limitations were identified, the Missile Tutor helped demonstrate that the MITT design was robust enough to support another technical training domain.

#### 6.2 MITT Writer Workshop

The MITT Writer workshop was held on February 7 & 8, 1991 in Atlanta, GA (Browning, et al., 1991). Eighteen people from the Air Force, Navy, and NASA attended the two-day workshop that was supported by 5 Galaxy Scientific personnel.

The workshop had the following goals:

- to provide an initial test of MITT Writer usability
- to teach participants to use the authoring system
- to prompt ideas for subsequent MITT Writer workshops

#### 6.2.1 Workshop Format

The workshop consisted of both formal presentations and extensive laboratory exercises. The formal presentations addressed a variety of concepts, including functional flow, problems, faults, alarms, display elements, and procedural editors. After each formal presentation, the groups were given directed exercises to complete. The participants were divided into small groups (3 people/group) for the laboratory exercises.

The demonstration system for the workshop was an Automobile Engine Tutor. The participants were given a tutor with portions omitted. The exercises were designed to aid users in completing the tutor. Also, each group was encouraged to customize their Automobile tutor.

Upon completion of the workshop, participants were asked to evaluate MITT Writer by completing a four-page questionnaire. The questionnaire covered three major topic areas: the workshop, MITT, and MITT Writer. The following section reports the results of the evaluation.

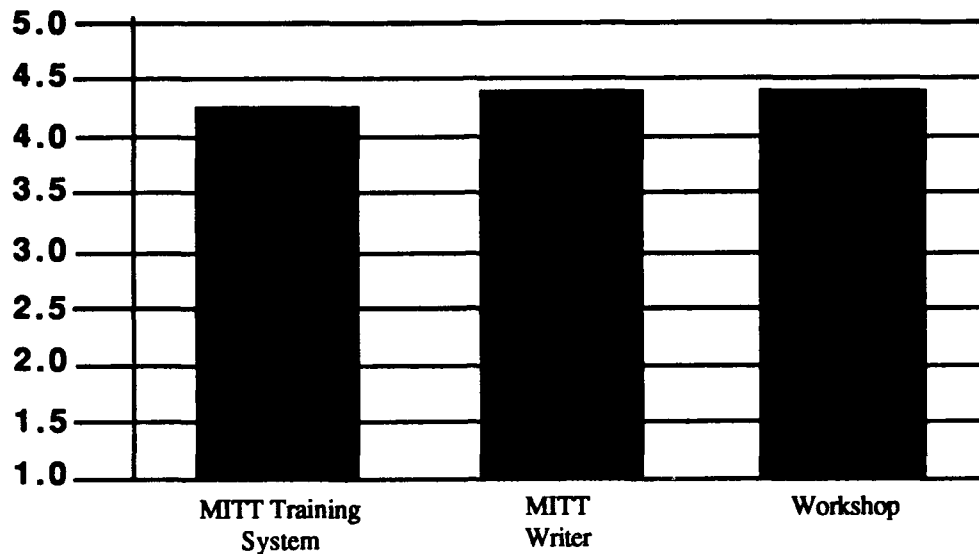
#### 6.2.2 Evaluation Results

Fourteen evaluation forms were returned at the end of the workshop. Respondents were asked to rate certain factors, using a 5-point scale: Excellent (5) to Unacceptable (1). The results from the major categories (Workshop, MITT, and MITT Writer) are summarized in Figure 17. Each of the categories received very high marks on the five-point scale. These average ratings are discussed below.

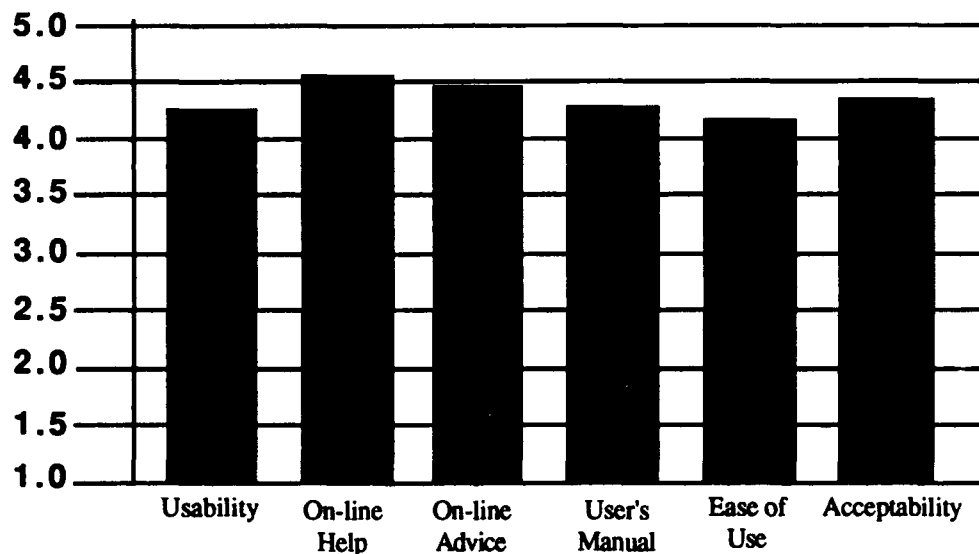
Without exception, the questionnaire respondents found the MITT training system easy to use (see Figure 18). When assistance or explanation was needed, the on-line help was available and useful. Respondents were very satisfied with the graphical appeal, clarity, and intuitive nature of MITT.

Respondents also liked MITT Writer (see Figure 19). The system has sufficient on-line help and advice to support the powerful authoring environment. Correspondingly, the highest rating on the evaluation was 4.8/5.0 for the MITT Writer on-line help.



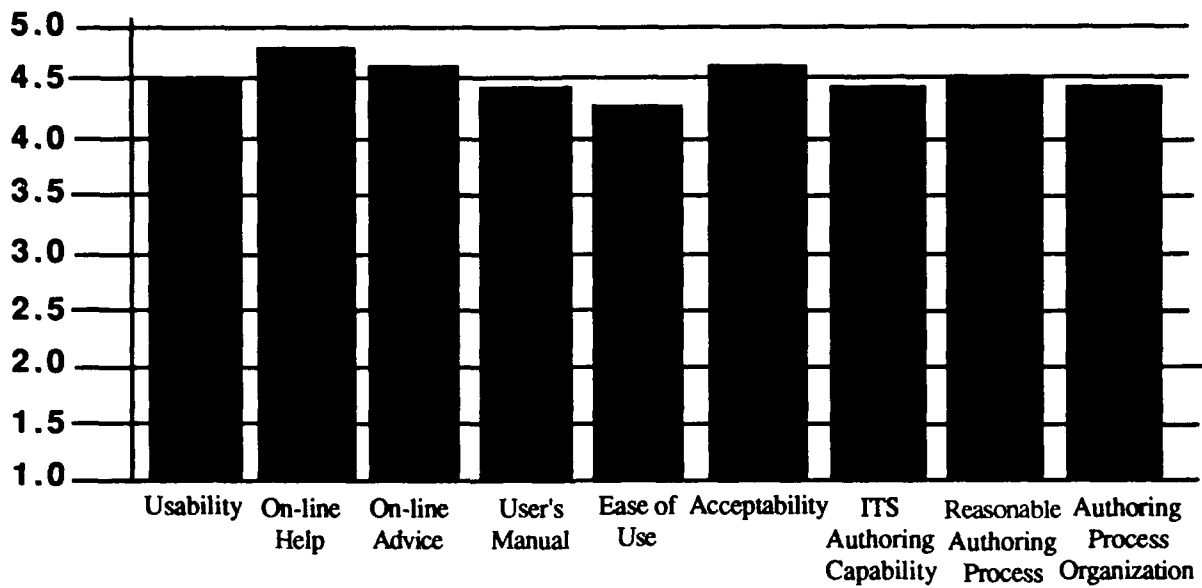


**Figure 17. Average ratings on major categories.**

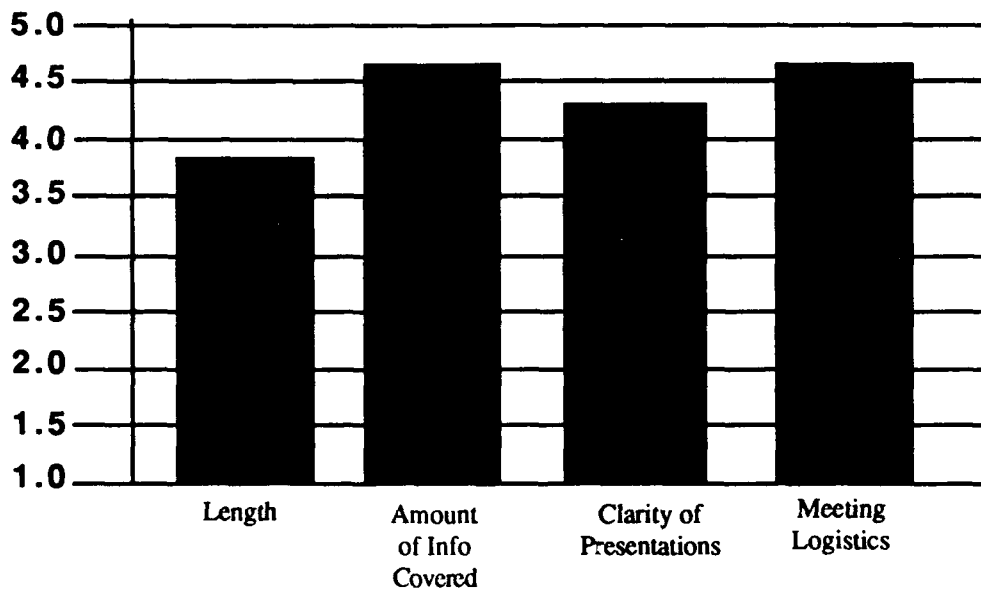


**Figure 18 MITT Training System scores - overall.**

Overall the workshop was a success (see Figure 20). It met the expectations of all respondents. However, a majority of the respondents were not completely satisfied with the length of the workshop. Two days did not seem to be long enough. As a result, this category received the lowest rating on the questionnaire - 3.9/5.0. Even though this rating falls in the "Good" range, the workshop feedback will influence planning for the next series of authoring system workshops.



**Figure 19. MITT Writer Ratings - overall.**



**Figure 20. Workshop Ratings.**

## **7.0 Conclusions and Areas for Future Development**

### **7.1 Conclusions**

The MITT and MITT Writer systems are a reasonable alternative for simulation-based diagnostic training. MITT tutors operate on IBM-compatible computer systems already installed in training installations, providing challenging, simulation-based problems to students. MITT delivers this training to students using ITS technology by presenting problems, monitoring student performance, comparing student actions to expert actions, and furnishing suitable feedback to the student.

MITT Writer provides economical development and maintenance costs for these training systems. It supports instructors and subject matter experts, rather than programmers, in developing MITT tutors. It provides enhanced ITS authoring capabilities, guiding the author in developing MITT tutors rapidly and maintaining them. By following the protocols in the MITT Writer authoring environment, technical personnel can organize their knowledge in a format that is aligned with the MITT architecture.

Without exception, the participants in the MITT Writer workshop found the MITT training system easy to use. Respondents were very satisfied with the graphical appeal, clarity, and intuitive nature of MITT. They indicated acceptance of MITT Writer's ITS authoring capability and found that MITT Writer has sufficient on-line help and advice to support the powerful authoring environment. They also indicated that the MITT Writer authoring process was both reasonable and properly organized.

The Minuteman Missile domain required extensive analysis and planning to incorporate the Missile system into the existing MITT design. ATC personnel pointed out areas where the design of the MITT interface could be changed to accommodate the domain's emphasis on procedures. While certain limitations were identified, the Minuteman Missile Tutor demonstrated that the MITT design is robust enough to support a variety of technical training domains.

### **7.2 Areas for Future Development**

MITT and MITT Writer present a promising approach to the development of technical training. Future enhancements can be made that will increase the effectiveness of both the authoring environment and the training it supports, and work can be conducted that will increase the use and acceptance of the MITT training approach.

Another MITT Writer workshop is being planned to teach the use and application of MITT Writer for ATC. Additional workshops would ease the transfer of the training technology to user commands. Both MITT and MITT Writer are being used to develop a new MITT

tutor for a new domain. Based on this new MITT tutor, or other current MITT tutor projects under development at the Armstrong Laboratory Human Resources Directorate Training Systems Division, a formal evaluation of training effectiveness and cost effectiveness could be conducted.

The current upgrade in the standard USAF computing environment offers new opportunities to increase the functionality of both MITT Writer and MITT. From a training technology point of view, enhancements in student modeling and instructor guidance can be made to include additional diagnostic techniques and new uses for the student model (e.g., intelligent problem generation, increased explanations by the procedural expert). The simulation component of MITT could also be enhanced by providing support for multiple component failures, creating deeper simulation models, and adding intelligent on-line functional flow diagrams.

On the authoring side, additional objects (e.g., audio/video for information displays) and object editors (e.g., interactive functional flow editor) can be added. The capabilities of the procedure editor can be enhanced to support more robust procedures (e.g., include conditional branches).

The MITT and MITT Writer environments can be partially or wholly integrated to reduce the time necessary to test changes made to the training system. For example, "preview" tools can be added which the author can use to try out new procedures, to view screens in the manner that students would view them, or to test functional advice on newly modified functional flow networks.

Networking support can be added to allow for connectivity between multiple authors of a training system, or to allow connections between instructor and student. An instructional station component can be added to allow instructors to monitor a student's problem solving session, subsequently offering advice or manipulating the simulation.

MITT and MITT Writer were developed in the C programming language. As the tools prepare for transition into the USAF training community, support for Ada language versions may be required. Many of the suggested embellishments could be made during this translation process.

## **8.0 References**

Browning, E.J., Johnson, W.B., Wiederholt, B.J., Norton, J.E., and Morgan, C.S. (1991). *Microcomputer Intelligence for Technical Training: Workshop Report and Software Completion Plan*. Atlanta, GA: Galaxy Scientific Corporation.

Johnson, W.B. (1981). *Computer simulations for fault diagnosis training: An empirical study of learning from simulation to live system performance*. (Doctoral Dissertation, University of

Illinois, 1981), (*Dissertation Abstracts International*, 41(11) 4625-A. (University Microfilms No. 8108555).

Johnson, W.B. (1987). Development and evaluation of simulation-oriented computer-based instruction for diagnostic training. In W.B. Rouse (Ed.), *Advances in man-machine systems research*: Vol. 3. Greenwich, CT: JAI Press, 99-125.

Johnson, W.B. (1988a). Developing expert system knowledge bases for technical training. In L.D. Massey, J. Psotka, and S.A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned* (pp 83-92). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Johnson, W.B. (1988b). Intelligent tutoring systems: If they are such good ideas, why aren't there more of them. *Proceedings of the 10th Annual Interservice/Industry Training Systems Conference*, Orlando, FL: The Industrial Security Association, 399-406.

Johnson, W.B. and Fath, J.L. (1983). Design and initial evaluation of a mixed-fidelity courseware for maintenance training. *Proceedings of the 27th Annual Meeting of the Human Factors Society* (pp 1017-1021). Norfolk, VA: Human Factors Society

Johnson, W.B. and Fath, J.L. (1984). *Implementation of a mixed-fidelity approach to maintenance training* (TR-661). Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences.

Johnson, W.B., Maddox, M.E., Rouse, W.B., & Kiel, G.C. (1985). *Diagnostic training for nuclear power plant personnel, volume 1: Courseware development* (EPRI NP-3829). Palo Alto, CA: Electric Power Research Institute.

Johnson, W.B., Neste, L.O., and Duncan, P.C. (1989). An authoring environment for intelligent tutoring systems. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*. Boston, MA, 761-765.

Johnson, W.B., Norton, J.E., and Duncan, P.E., and Hunt, R.M. (1988). *Development and demonstration of an intelligent tutoring system for technical training (MITT)* (AFHRL-TP-88-8). Brooks AFB, TX: The Air Force Human Resources Laboratory.

Johnson, W.B., and Rouse, W.B. (1981). Analysis and classification of human errors in troubleshooting live aircraft power plants. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-12, (3), 389-393.

Johnson, W.B., Wiederholt, B.J., and Maddox, M.E. (1986). *Diagnostic training demonstration: Instructor and student manuals and sample diskettes* (EPRI NP-4493P). Palo Alto, CA: Electric Power Research Institute.

Maddox, M.E., Johnson, W.B., & Frey, P.R. (1986). *Diagnostic training for nuclear power plant personnel, volume 2: Implementation and evaluation* (EPRI NP-3829-II). Palo Alto, CA: Electric Power Research Institute.

Massey, L.D., Psotka, J., Mutter, S.A. (Eds.) (1988), *Intelligent Tutoring Systems: Lessons Learned*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Neste, L.O. (1989). Overcoming microcomputer constraints in the development of an intelligent tutoring system. *Fall Symposium on Computing Research and Development*. Houston, TX: University of Houston Clear Lake, Research Institute for the Computing and Information Sciences.

Norton, J.E., Wiederholt, B.J., and Johnson, W.B. (1991). *Microcomputer Intelligence for Technical Training (MITT): The Evolution of an Intelligent Tutoring System* (1991). Atlanta, GA: Galaxy Scientific Corporation.

Polson, M.C., and Richardson, J.J. (Eds.) (1988). *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Rouse, W.B. (1979). Problem solving performance of first semester maintenance trainees in two fault diagnostic tasks. *Human Factors*, 21(5), 611-618.

Rouse, W.B., and Hunt, R.M. (1984). Human problem solving in fault diagnosis tasks. In W.B. Rouse (Ed.), *Advances in Man-Machine Systems Research: Vol 1*. Greenwich, CT: JAI Press, 195-222.

Wiederholt, B.J. (1991). *MITT Writer User's Reference Manual*. Atlanta, GA: Galaxy Scientific Corporation.